Autonomous Building Lighting Assessment

Dr. Bryan Rasmussen

ENGR 491 - 520

Team Members:
Randy Ardywibowo
Ramsey Bissex
Nicolas Botello
Amir Darwesh
Hongxiang “Casper” Fu
Blake Karwoski
Jusung Lee
Garrison Neel
Kevin Patel
Vince Toledo
Yang Yang

Table of Contents

List of Tables

List of Figures

Introduction

Written by: Randy Ardywibowo, Nicolas Botello

An energy audit is an analysis of energy usage in a building in order to maximize the energy efficiency of a building. An important part of this audit is the light audit, the inspection and analysis of lighting types and positions to provide adequate light levels in a building while minimizing energy consumption. This type of energy audit is important, as over 30% of the energy usage in industrial buildings is derived from lighting and HVAC systems [1]. Although important, these energy audits are time consuming. Detailed light audits are particularly difficult as they require gathering and analyzing spectrometer readings of each light in a building. Therefore, we propose to develop a device which simplifies light audits.

This project's aim is to develop a commercially viable device capable of autonomously classifying, enumerating, and evaluating building lighting systems for the purposes of conducting energy audits. To perform an autonomous light audit, our proposed device will need to map a building in real-time as well as identify the location and type of lights. Recognizing these two tasks, the project is divided into two teams: one in-charge of developing the mapping algorithm, and another in-charge of lighting identification.

The light identification team is charge of developing the lighting identification and analysis algorithms. This task can be divided into three subtasks: detecting lights, determining light locations, and taking spectrometer readings of each of the detected lights. To accomplish these tasks, our device will require a camera and brightness sensor to detect lights, a distance sensor to aid in determining light locations, and a spectrometer to take spectrometer readings of lights.

On the other hand, the mapping team is in charge of creating a 2D map of the building being audited. This involves being able to create dynamic 2D maps with the use of a scanning device and an inertial measurement unit (IMU) sensor. In order to create these maps we will be using a package from Robot Operating System (ROS) framework named hector_slam that will do most of the work. So the tasks will be divided into deciding into which scanning device to use to detect the environment, using sensor with hector_slam to create 2D map, and IMU integration for more precise location.

Knowing our device's specifications, we propose to accomplish the following tasks. Firstly, we need to demonstrate the ability to use each of these sensors. Secondly, we need to integrate them all through the Robot Operating System (ROS) framework. After integrating the sensors into ROS, A light audit is the analysis and subsequent optimization of lighting types and positions in order to maximize the energy efficiency of a building's lighting infrastructure; it is a large part of energy benchmarking, which helps to understand energy usage and cut waste. This is an especially important process for businesses due to the sheer numbers and size of business offices and buildings, where even the smallest relative optimization can mean a significant cut in operating costs.

Background

Written by: Randy Ardywibowo

A light audit is the analysis and subsequent optimization of lighting types and positions in order to maximize the energy efficiency of a building's lighting infrastructure; it is a large part of energy benchmarking, which helps to understand energy usage and cut waste. This is an especially important process for businesses due to the sheer numbers and size of business offices and buildings, where even the smallest relative optimization can mean a significant cut in operating costs.

Problem Statement

Detailed lighting audits are time consuming and complex; an autonomous light-auditing system is necessary to answer these challenging constraints. Such a system must be capable of identifying the location, and specification of lights in an industrial setting, where the implementation and usage of the system does not exceed the cost of traditional, manual light auditing; in addition, the process with the new system should be faster, and the controller should be relatively easy to learn and use.

Solution

Written by: Randy Ardywibowo, Ramsey Bissex

To solve this problem, we created a robot which would autonomously conduct this light audit. The robot has two major functionalities. First, the robot maps the building and localizes the robot's position in the building. Second, the robot gathers the brightness levels of the building and finds lights located in it. After finding the lights, the robot is able to analyze each light's type based on its emitted light spectrum and its dimensions. With these two datasets gathered, a building lighting optimization program is able to process these datasets and generate a report of potential optimizations on the building's lighting plan.

To accomplish each of these tasks, numerous sensors need to be used. A camera is required to detect lights and measure their dimensions, a laser distance sensor is needed to map the building, a brightness sensor is needed to gather brightness data, a spectrometer is essential in measuring the emitted spectrum of lights, and finally another distance sensor is needed to locate lights in three dimensional space. These fragile sensors require adequate enclosures to prevent any damage caused to them.

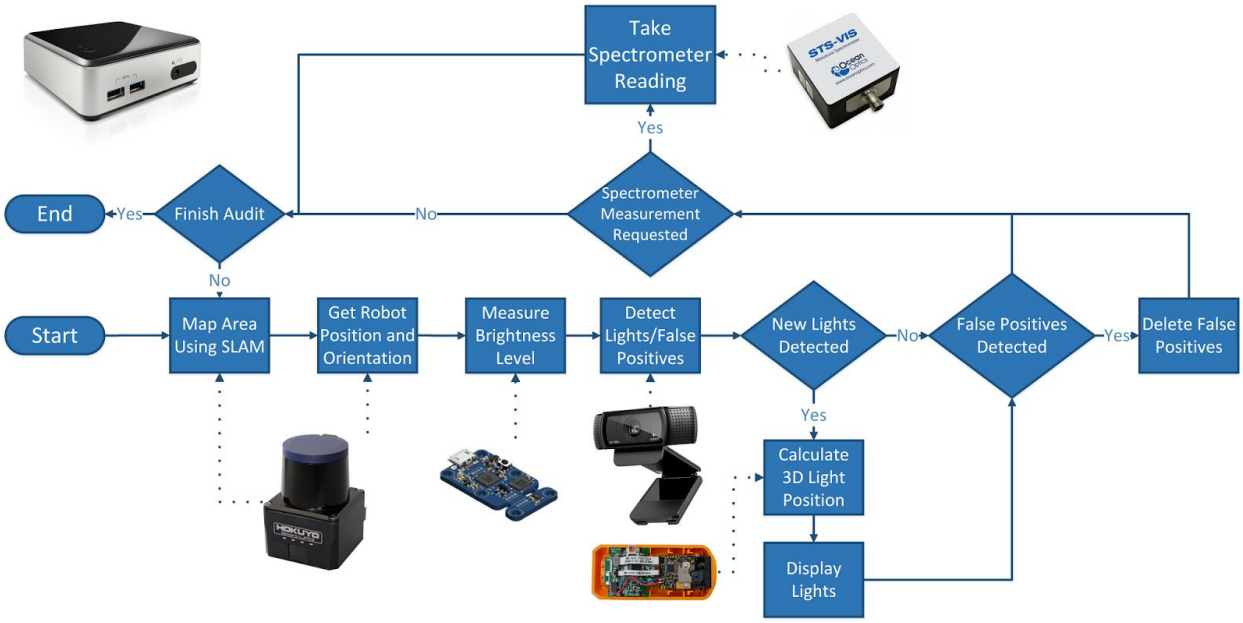


Figure 1: Flowchart of the Robot's Building Light Auditing Process

Hardware

Computing Platform

Written by: Randy Ardywibowo

The computing platform we chose for our project was the Intel® NUC Kit D54250WYK. We chose this platform for several reasons. Firstly, it provides the necessary computing power needed for our computing intensive tasks. These tasks include mapping and localization of the building and robot respectively, image stream processing at a high refresh rate, image object detection, and geometric object manipulation.

The second reason why we chose this platform is because it has a low power consumption when compared to other desktop computing platforms. With a load power consumption at lower than 35W [25], this platform is optimal for our purposes when compared with other platforms which typically goes above 100W [25].

Finally, the platform has a small and light form factor, with an exterior dimension of 116.6 x 112.0 x 34.5 mm and a total weight of 478 g [25]. This allows the platform to be mounted on the robot without adding any major issues concerning the product's overall weight.

Figure 2: Intel® NUC Kit D54250WYK [25]

Camera

Written by: Vince Toledo, Garrison Neel

The camera used in this system is the Logitech C920. The C920 is a USB webcam with a maximum resolution of 1920x1080 and maximum framerate of 60 frames per second [2]. Contrast, exposure, aperture, ISO, and other settings are manipulated before recording using v4l2ctrl, a linux driver control application. The image manipulation allows for lights to be more clearly bounded by creating high contrast between the lights and their surroundings. The image settings of importance are assigning an exposure of 3ms and a gain of 100.

The C920 is used in conjunction with the distance sensor to determine light locations in 3D space. OpenCV is employed in order to generate the bounding boxes of each light.

Figure 3: Logitech C920 [2]

Spectrometer

Written by: Randy Ardywibowo

The spectrometer we used was the STS-VIS Miniature Spectrometer made by Ocean Optics. This spectrometer has a small size compared to other spectrometers, with dimensions of 40x42x24 mm [3]. This can be seen in Figure 4, where the spectrometer's size is compared with a coin. Moreover, the spectrometer is connected to our package via USB connection. These specifications make the spectrometer suitable for our package, as it needs to be small enough for our robot mounted configuration, and it is easily connected using USB connection.

This spectrometer was interfaced with OmniDriver, a driver made by Ocean Optics. This driver provides useful controls for the spectrometer such as controlling its integration time, scan amount, and detecting whether its output is saturated due to high input light intensities [3]. The driver outputs the spectrometer readings as 1024 intensity values for different wavelengths. These wavelengths range from 350-800 nm.

Figure 4: STS-VIS [3]

Distance Sensor

Written by: Vince Toledo

In order to locate light locations in 3D space, our group used the Porcupine Labs LR4 Laser Distance Sensor. The distance sensor was chosen for its performance relative to other distance sensors in its price range. With a range of 50 meters, accuracy within 3mm, and USB compatibility, the laser scanner was the perfect fit for our project [4]. Unfortunately, the 2.5 Hz measurement rate of the device was only demonstrated in ideal conditions. When faced with reflective panels or longer distances, the sensor would become the bottleneck of the combined system. The sensor's reliability was also questionable due to hardware issues appearing at solder points and ribbon contacts. Limited driver support for Linux provided further issues. Future teams would benefit from an industrial grade distance sensor to replace the current one, or use of a stereoscopic camera arrangement for height readings.

All of this being taken into account, the distance sensor served its purpose for our application. We were able to successfully integrate the manufacturer-provided drivers into the ROS

environment, and publish the data on a ROS topic. The sensor provided reliable distance measurements to within a 1-2 millimeters in our testing.

Figure 5: Porcupine Labs LR4 [4]

Brightness Sensor

Written by: Randy Ardywibowo, Ramsey Bissex

To measure the brightness levels of a building, we chose the Yocto-Light V3 by Yoctopuce. This brightness sensor is the same chip used in many phones to detect brightness so that the phone screen brightness can be set correctly. This sensor has a spectral sensitivity of 400nm-750nm which is slightly better than the human eye [5]. The sensor has a length and width of 20 x 35.5 mm. It measures brightness in lux and has a sensitivity of 0.25 lux with a 4 Hz refresh rate. It can read brightness levels from 0 to 100,000 lux.

To interface with this device, we used the driver library provided by the manufacturer for the C++ programming language. This library is provided without any added costs and is available for different operating systems such as Windows, Linux, and Mac OS X. The library provides basic features to take brightness readings and error handling methods.

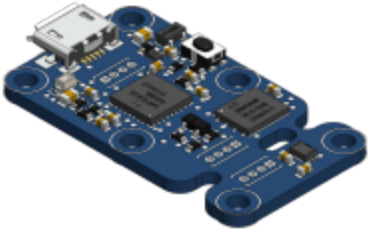


Figure 6: Yocto-Light V3 [5]

Case

Written by: Blake Karwoski

A case was designed to house all of the sensors used, and to assist in their cable management. The case was designed in Solidworks. The initial case, which was printed in black ABS, held the sensors fairly well, and was designed to eventually become a part of a mobile, handheld platform.

Figure 7: First case iteration, as used for the first demonstration

This case needed to be revised because the new laser scanner had slightly different dimensions than the old one, such as the location of the power and data cables. The distance sensor also needed a new housing. The original housing for this sensor was bulky and impractical for the final device, so the essential components (circuit board and optical components) were removed. These components were trickier to design a case for, so this section of the case was created separately. This allowed multiple drafts of the enclosure to be created with minimal wasted plastic and print time. An example of an intermediate stage design, printed with white ABS, is shown below.

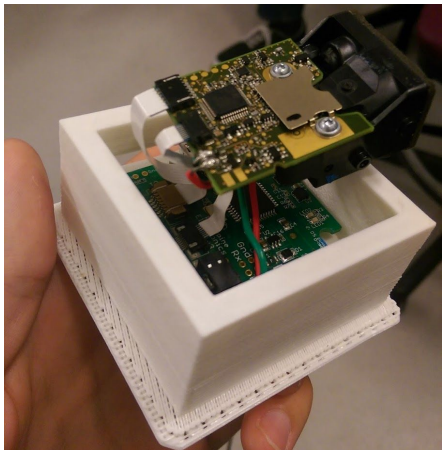


Figure 8: Separately designed & printed distance sensor enclosure

This component was designed to attach to the main portion of the base with #4-40 screws. The intermediate case was printed in two pieces with light blue ABS, on the flashforge printers at the EIC.

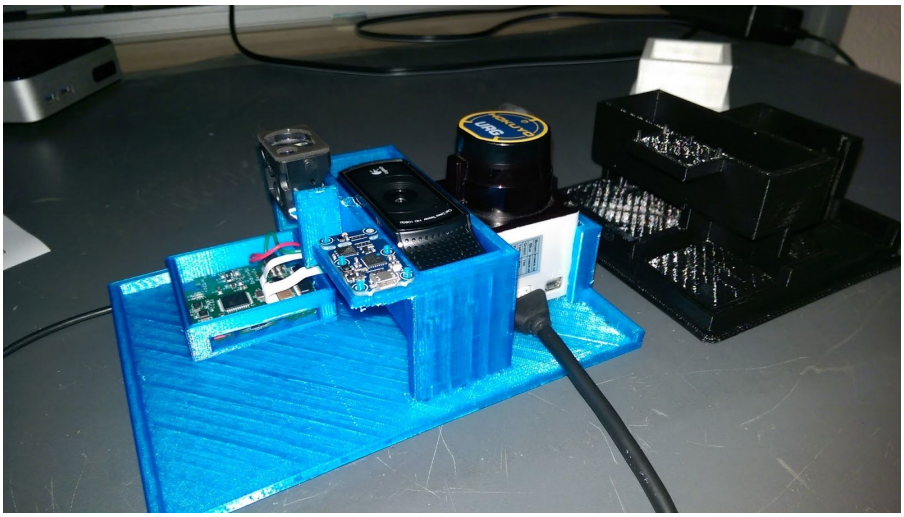


Figure 9: Second case design (left), shown on the EIC bench with the first case design (right) and the distance sensor enclosure (back right)

This case had solved most of the pre-existing issues, but there were some difficulties routing the laser scanner cable through the back of the device, and the intended position of the fiber optic cable would have been within the field of view of the laser scanner. These issues were all resolved with a final iteration of the case.

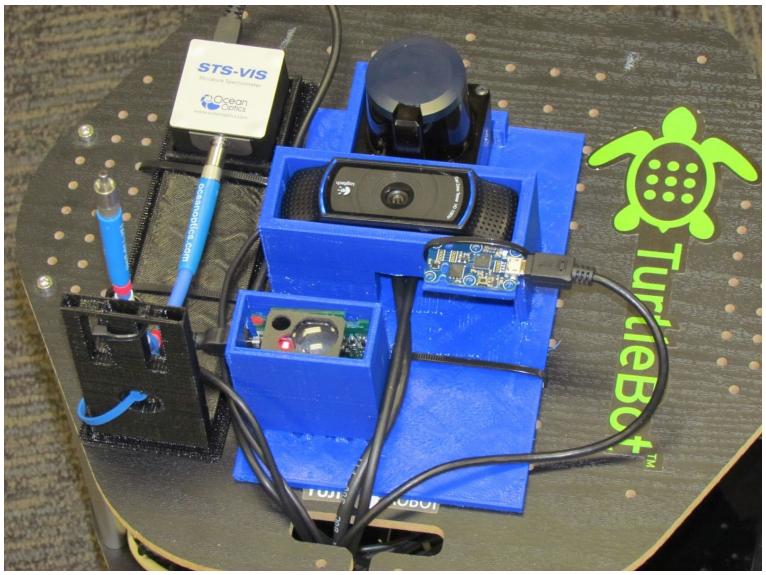


Figure 10: Final case configuration, as used for the device demonstration at the MEOB.

Each of the components housed by the case posed a unique set of challenges to work around. The laser scanner, perhaps the most challenging, had a field of view of 240° horizontally that

could not be obstructed. The webcam, brightness sensor, and distance sensor each required an unobstructed view of the ceiling. The spectrometer's fiber optic cable also had to be aimed at the ceiling. It had a minimum bend radius of a few inches and lacked rigidity, so a dedicated support post was created to hold it vertically. Finally, every component had one or more cables for data and power that needed to be accounted for.

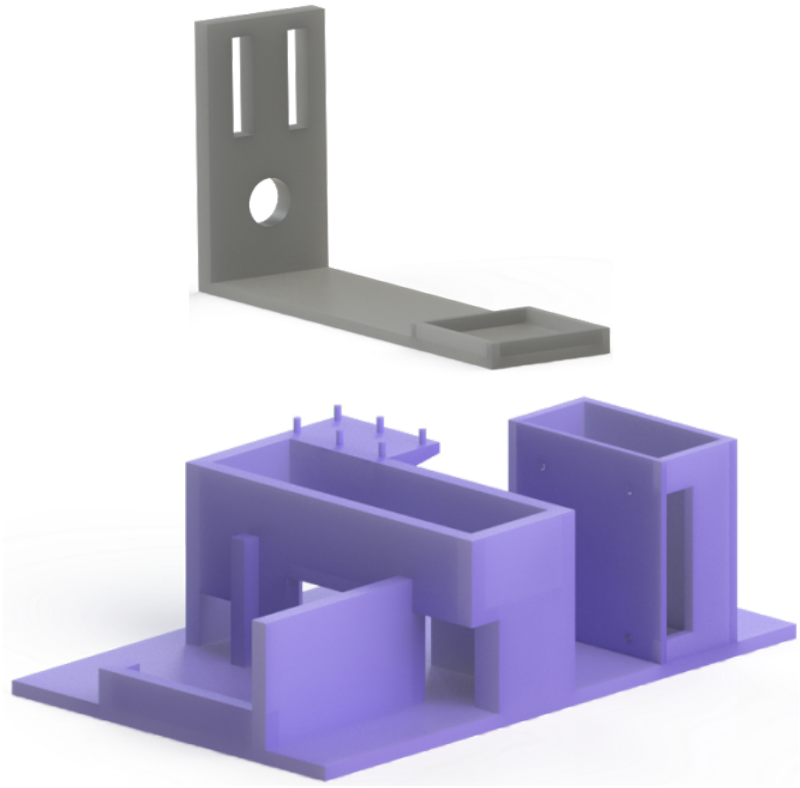


Figure 11: Solidworks renderings of the components of the case design.

Kinect

Written by: Yang Yang

The reason we use Kinect is for its color and depth-sensing cameras, which have a horizontal field of view of 57 degree and data stream of 30 frames per second. We planned to using the kinect can generate a textured map of the environment.

Figure 12: Kinect[16]

Laser Scanner

Written by: Nicolas Botello

The laser scanner we propose to use is the Hokuyo-URG-04LX by RobotShop.It is a laser scanner that has a reach of about ten meters and is able to detect about 240° area scanning range with 0.36° angular resolution with a 100 ms scan speed. We believe that the capabilities of this model of the laser scanner will be able to detect the environment.

Figure 13: The Hokuyo URG-04LX Laser scanner [17]

Inertial Measurement Unit (IMU)

Written by: Nicolas Botello

The Inertial measurement unit (IMU) sensor we propose to use is the Razor IMU from SparkFun. We choose to use this sensor not only because of its small size and low cost, but it is able to retrieve 9 degrees of freedom which come from the following sensors onboard 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer. We believe that we are going to need

an IMU to be able to find the pivot and position of the device with more precision.To retrieve the information from this IMU sensor, we will use the ROS package called razor_imu_9dof.

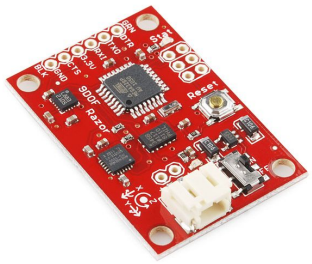


Figure 14: The 9 Degrees of Freedom - Razor IMU [19]

Turtlebot

Written by: Nicolas Botello

Figure 15: Turtlebot [20]

The unmanned ground vehicle we proposed to use was the Turtlebot by Yujin Robotics, We proposed to use this because of its low cost and open source software. It is a robot that is usually used for a lot of home projects and has the capability of being able to driven via teleop. It at the same time comes with the kinect and is able to drive and visualize where it is driving using a 3D point cloud.

Router

We proposed to use the TP-Link router, because of its low cost, 300Mbps connection speed, and multiple ethernet ports. We propose to use this router, so that we are able to dedicate a specific network to connect remotely to the device and not have any issues with any lag.

Figure 16: TP-Link [22]

Tablet

Written by: Kevin Patel

The tablet we propose to use is the Google Tango Tablet. We choose this tablet for the fact that it has built in developer resources that making connecting to computer or the turtlebot very easy. The tango as well has a Motion Tracking Camera and 4MP 2µm Pixel Camera, using these two cameras the tablet is able integrate depth sensing. This depth allows the tango to track motion, depth Perception in 3D. While we did not use these features, they were very helpful in testing how to map. These features may even help us with future developments in mapping differently in more environments.

Figure 17: Google Tango[21]

Light Detection and Location

Written by: Randy Ardywibowo

This section describes the method we used to detect and locate lights. This method was used both for finding the position of light fixtures in 3D space, as well as calculating the dimensions of bulbs within each respective fixture. This task is divided into several parts. Firstly, the camera which detects lights is calibrated using the camera calibration package available in ROS. Second, the lights are found in the image using OpenCV, a computer vision library. Third, the lights found in the 2D image is projected as rays in 3D space. Finally, using data from a distance sensor, the 3D light location is found by finding the point of intersection of the light with a ceiling plane. By changing the thresholding parameters, this method was also used to calculate bulb dimensions. This process is illustrated by the flowchart in Figure 18.

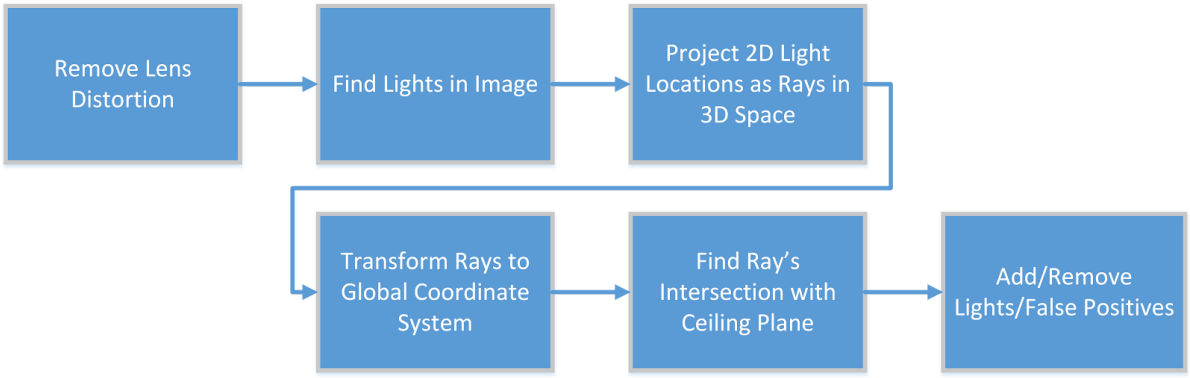


Figure 18: Flow Chart of Light Detection and Location Method

Camera Calibration

Written by: Randy Ardywibowo

In order to find the location of lights in real world units of length, the camera stream is first calibrated to remove any distortion inherent in the camera lens. This process estimates the parameters of the lens and image sensor of the camera, including camera rectification, projection, and distortion. By saving these parameters to a file, the camera image can be transformed to negate this undesirable lens distortion. This allows our method to process the output of the camera image stream using the pinhole camera model [23].

The camera calibration was done using the camera_calibration package available in ROS. To accomplish this process, a checkerboard pattern was printed and laid on a flat surface. The camera_calibration package was then run using the Logitech C920 camera. Pointing the camera at the checkerboard pattern, the package is able to detect the vertices of each square on the checkerboard. Finally, the package calculates the distortion parameters of the camera and saves them to a text file which can be read by another program. This process is shown in Figure 19.

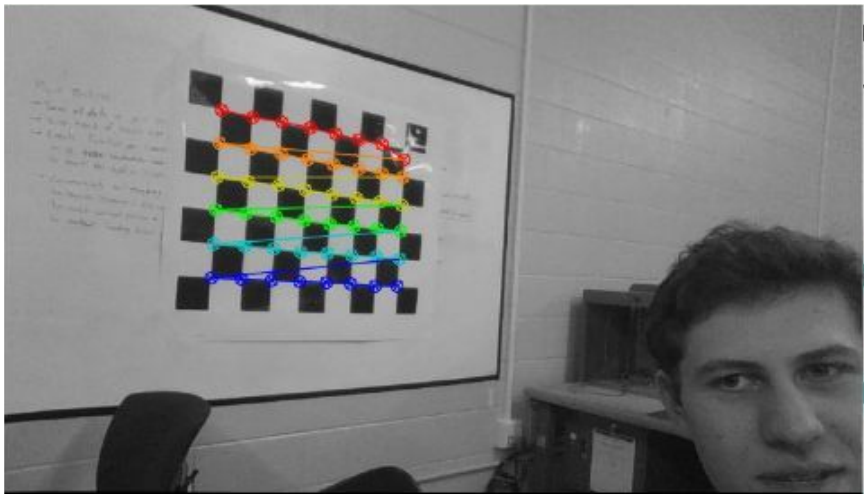


Figure 19: Camera Calibration Method

Light Detection

Written by: Garrison Neel, Randy Ardywibowo

The normalized images received from the webcam are firstly converted to grayscale, thresholded, and morphed. The lights are then detected using the contour detection feature in OpenCV. This powerful function allows us to detect various properties of the light such as its midpoint, bounding rectangle, and size in the image. As a visual aid and debugging tool, bounding boxes are drawn on the lights, as shown in Figure 20. When detecting bulbs and calculating its dimensions, a similar process is used. This process uses stricter tolerances and a more sensitive threshold so that the only detected object in the image is the bulbs. This bulb detection method is shown in Figure 21.

Figure 20: Light Fixture Detection

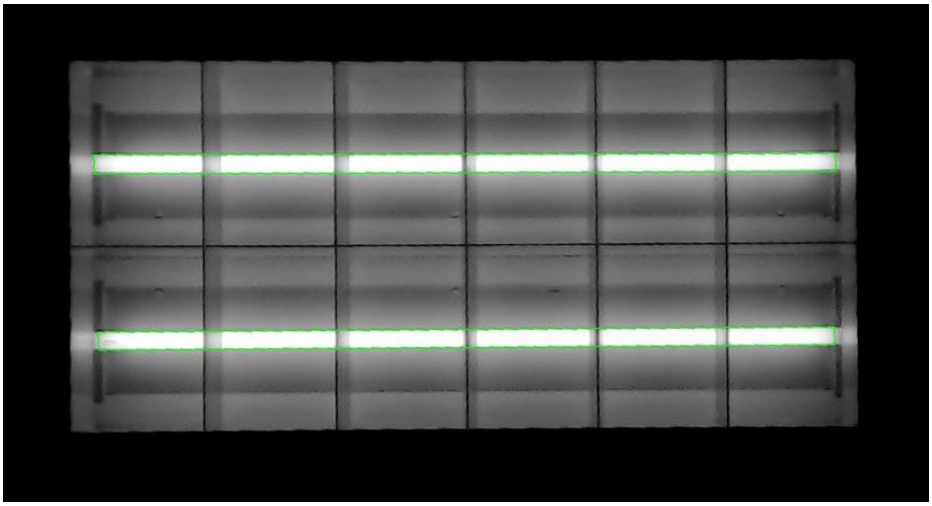


Figure 21: Bulb Detection with Stricter Color Thresholding

Otsu thresholding is used to find the bulbs in the image. Otsu thresholding adaptively thresholds to the highest peak in bimodal histograms. Since bulbs are typically brighter than the remainder of the image, Otsu thresholding works well. A diagram of this is shown below in Figure 22.

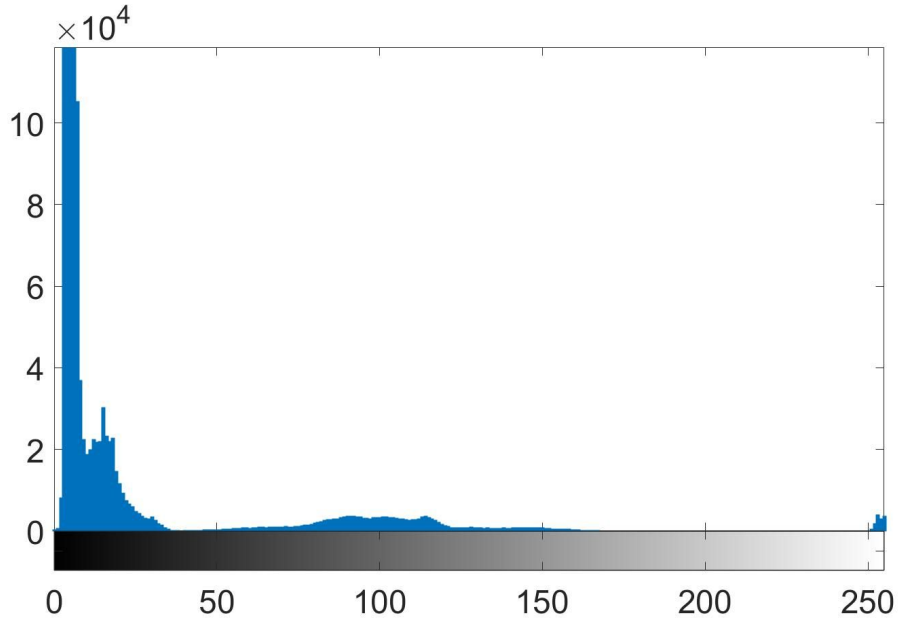


Figure 22: Histogram of Light Detection Based on Light Size and Color

The peak on the right hand side represents the bulbs in the image, and is easily found with Otsu thresholding. One caveat is that Otsu alone is not enough to detect if there is a light in the image. If Otsu thresholding is used on a black or dark image, it will falsely detect lights.

3D Light Location Method

Written by: Randy Ardywibowo

After receiving the pixel coordinates of a light from the light detection algorithm, we can calculate the position of a light in three dimensional space using inverse camera projection algorithms. To simplify this process, a camera coordinate system was used. This coordinate system treats the line of sight of the camera as always pointing in the x-direction and the camera always being at the origin. This coordinate system is illustrated in Figure 23.

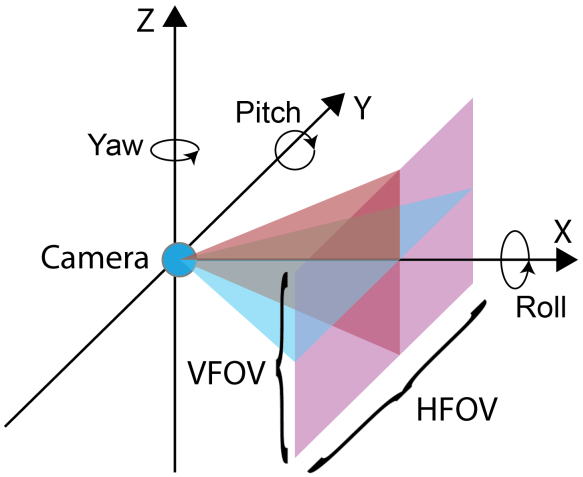


Figure 23: Graph of Camera Coordinate System, Rotation Angles, and Field of View

Because lights are found in a 2D image, some information on the 3D position of the lights is lost. This causes pixels in the image to be represented not as points in 3D space, but as rays or lines as shown in Figure 24. The transformation of pixel location to lines in 3D space is derived on the following page.

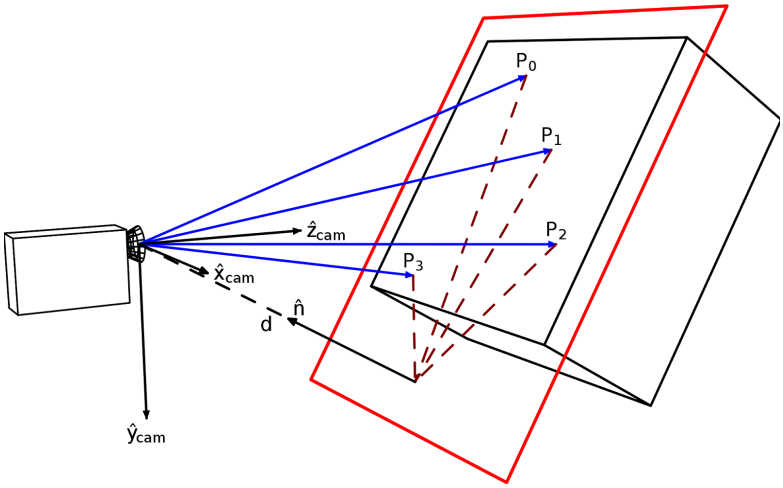


Figure 24: Camera Projection Illustrated in Three Dimensions

This is the basis of the pinhole camera model, which can be applied as the camera image stream has been normalized with the camera calibration process. The illustration above is simplified into two dimensions in Figure 25. From the graph in Figure 25, we see that with the pinhole camera model, the light angle and FOV/2 share a common side. Using this information, the following equation which relates the angles with the 2D image location of the light can be derived:

$$\frac{\tan \alpha_{light}}{\tan\left(\frac{FOV}{2}\right)} = \frac{L_{light}}{\left(\frac{L_{image}}{2}\right)}$$

Solving for the angle α_{light}, This equation can be transformed into the following equation.

$$\alpha_{light} = \tan^{-1}\left(\tan\left(\frac{FOV}{2}\right) * \frac{L_{light}}{\left(\frac{L_{image}}{2}\right)}\right)$$

Solving this equation for both horizontal and vertical fields of view allows us to project light pixels into rays in 3D space.

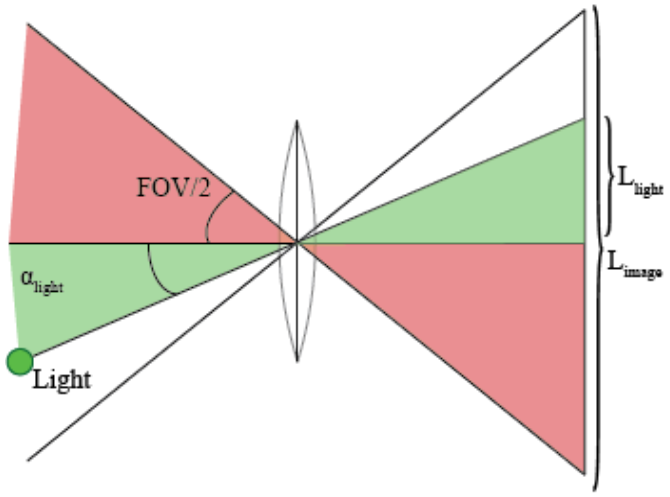


Figure 25: Camera Projection Illustrated in Two Dimensions

After the light pixels are projected as rays in 3D space, the rays can be transformed from the camera coordinate system, to the global coordinate system. This transformation involves both rotation and translation. The transformation was done using Tait-Bryan angles roll (θ_R), pitch (θ_P), and yaw (θ_Y). Because the robot rotation data received from Hector SLAM is in quaternion

rotation (q_0, q_1, q_2, q_3) instead of Tait-Bryan angles, the rotation data must first be converted. This is done using the following formula [24]:

$$\begin{pmatrix}\theta_R\\ \theta_P\\ \theta_Y\end{pmatrix} = \begin{pmatrix}\arctan\dfrac{2(q_0q_1+q_2q_3)}{1-2({q_1}^2+{q_2}^2)}\\ \arcsin(2(q_0q_2-q_3q_1))\\ \arctan\dfrac{2(q_0q_3+q_1q_2)}{1-2({q_2}^2+{q_3}^2)}\end{pmatrix}$$

With the coordinate system defined for the camera above, the rotation matrices for roll R(θ_R), pitch P(θ_P), and yaw Y(θ_Y), are defined as follows: roll is rotation about the x-axis, pitch is rotation on y-axis, and yaw is rotation on the z-axis. With these definitions, the rotation matrix for each angle is defined by the following:

$$R(\theta_R) = \begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\theta_R) & -\sin(\theta_R)\\ 0 & \sin(\theta_R) & \cos(\theta_R)\end{pmatrix}$$

$$P(\theta_P) = \begin{pmatrix}\cos(\theta_P) & 0 & \sin(\theta_P)\\ 0 & 1 & 0\\ -\sin(\theta_P) & 0 & cos(\theta_P)\end{pmatrix}$$

$$Y(\theta_Y) = \begin{pmatrix}\cos(\theta_Y) & -\sin(\theta_Y) & 0\\ \sin(\theta_Y) & \cos(\theta_Y) & 0\\ 0 & 0 & 1\end{pmatrix}$$

It is important to note the order that these rotations are performed. From the definition of Tait-Bryan angles, in order to rotate objects from the camera coordinate system to the global coordinate system, roll rotation must be done first, then pitch, and finally yaw [24]. After these rotations are performed, the object can then be translated to the global coordinate system by adding the robot's position (X_O) to the rotated object.

$$X_{global} = Y(\theta_Y)P(\theta_P)R(\theta_R)X_{camera} + X_o$$

With this formula, the equation to transform objects from the global coordinate system to the camera coordinate system is as follows:

$$X_{camera} = R(-\theta_R)P(-\theta_P)Y(-\theta_Y)(X_{global} - X_o)$$

Note that the equation above was simplified by realizing that the inverse of a rotation matrix is found by negating the rotation angle.

With these equations, any geometric object can be described in the camera coordinate system to then be transformed into the global coordinate system and vice versa. In particular, the rays generated from the 2D light location can be transformed into the global coordinate system.

To find the light's 3D light position, a distance sensor measuring the robot's distance to the ceiling was measured. This distance reading was used to create a ceiling plane at the height measured. With this, the 3D light location algorithm was calculated by finding the light ray's intersection with the ceiling plane. This was done by solving the following system of equations:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix} + \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} t$$

$$Ax + By + Cz = D$$

Where the top and bottom equations represent equations for a line and a plane respectively. The solution to this equation is a point in 3D space which represents the location of the light.

In creating this light location projection algorithm, several additional geometric functions were also created. These were created to simplify geometric calculations. These functions calculate useful information such as distance, point of intersection, point of minimum distance, midpoints, dot products, cross products, and magnitudes of various geometric objects such as lines, planes, vectors, and points.

Several critical cases arise when using this method. These cases include when lights are on the edge of the camera image, when position data received from Hector SLAM is inaccurate, and when any network issues may cause errors in light detection. To mitigate these problems, proper false positive detection, thresholding, and light position update techniques are performed.

Light Position Update

Written by: Randy Ardywibowo

Because lights seen in the image are blobs instead of single pixel points in the image, inaccurate light positioning may occur when these lights are seen on the edge of the camera image. This is because the light detection algorithm done by OpenCV does not take into account that lights may extend out of the image's field of view. Because of this, the lights detection treats the midpoint of the light as the midpoint seen in the image, instead of the actual light midpoint which may extend out of the camera's view. This problem is illustrated in Figure 26. In this figure, the light has been detected by OpenCV, however, it is not fully within the camera's field of view, causing a slight error in finding its midpoint.

To mitigate this, lights on the edge of the image are marked, noting that the full light fixture has not been seen fully by the camera image. This is done by checking if the vertices of the light

are within a certain threshold of the image's edge. After the full light fixture is in the camera's field of view, the midpoint of the light is recalculated and updated in the map's display.

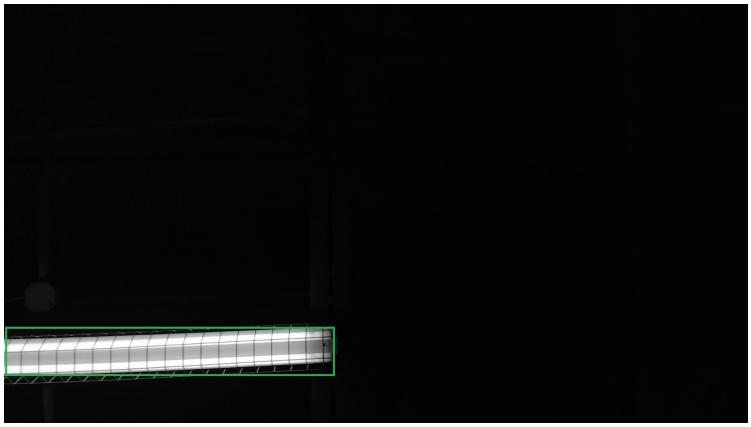


Figure 26: Light Seen Partially by the Camera

False Positive Detection

Written by: Randy Ardywibowo

To mitigate any errors in inaccurate pose data from Hector SLAM, lights previously found by the light location algorithm are continuously compared with lights seen in the camera's field of view. With this, lights displayed in the map but are not detected by the camera are deleted. This process is illustrated in Figure 27.

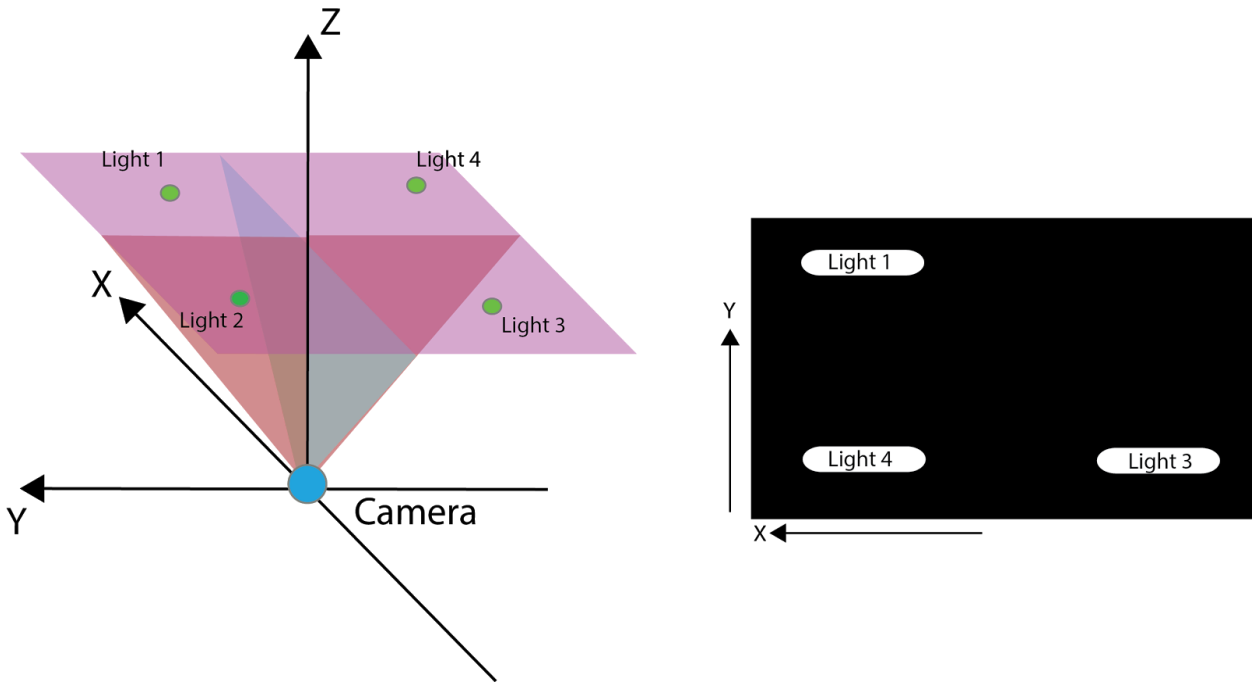


Figure 27: Illustration of Light False Positive Detection

Shown in Figure 27, light #2 was previously detected and placed in the map. However, despite being within the camera's field of view, the light is nowhere to be seen. Because of this, light #2 will get deleted from the map.

Bulb Dimension Calculation

Written by: Randy Ardywibowo

Bulb dimension calculation is done similar to light detection and location. However, instead of finding the 3D position of the light's midpoint, this method find the 3D position of each bulb's vertices. Using this position data, the length and width of each bulb can be calculated by finding each vertex's distance to one another.

Position Thresholding Method

Written by: Randy Ardywibowo

To mitigate further false positives due to inaccurate position data and partially seen lights, lights found are thresholded based on their distance to existing lights. This is done by finding the newly found light's distance to existing lights. If this distance is found to be within 0.5 meters apart, the newly found light is disregarded and treated as being the same as a light that was previously found.

Light Analysis

Spectrometer data can be compared to reference data to figure out light type. Bulb size is analyzed using a combination of the 3D algorithm and OpenCV. Bulbs are associated with a light and then all data (position, bulb size, light type, light ID) are written to file.

Mapping Package

Written by: Nicolas Botello

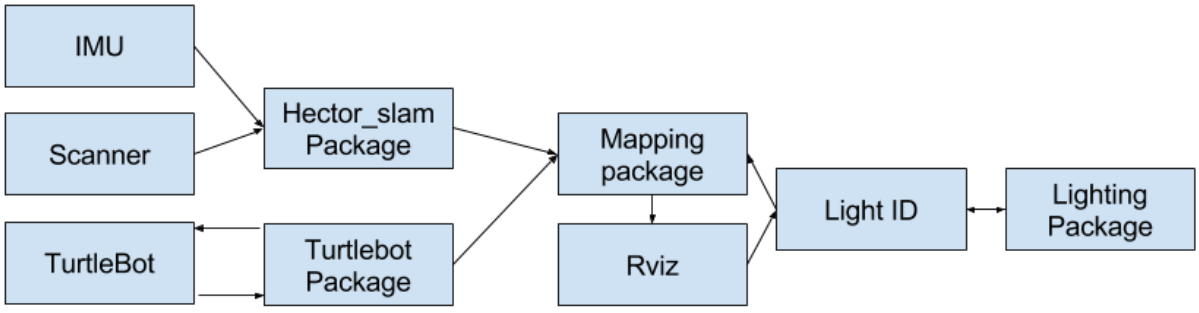


Figure 28: Diagram representation of all components

The mapping package named uprobotics is a package that utilizes the urg_node to be able to retrieve data from the hokuyo scanner and sends the data into the hector_slam node to convert that to an actual map. It then utilizes Rviz to show the creation of the map and displays the lights when detect and if you press on a light it will then send a request to the lighting team to take a picture and detect more detail information on that light and store it. It utilizes the turtlebot as a device to sit on and map with no movement in the z axes using a plain field at all times. The turtlebot can then be controlled using the keyboard or if commented out the launch file for keyboard teleop.

You are able control it using the robo recon mobile app if you are in the same network and connect to the ROS_MASTER_URI. We found that if you have teleop running to be able to control from keyboard as well as connecting to the topic from the tablet a twitching movement happens.

This package can be found at https://github.tamu.edu/bote795/KillerBeeMapping , if you look in the wiki you can find helpful information on how things work. The mapping package is mainly made up of launch files which ROS uses to be able to execute files without having to open multiple tabs. The only file that contains code that was written by the group is 'button.cpp' though most of the file is made out of information found from tutorials online and that file is used to create the lights in Rviz and receives the requests that are send to it when you click on a light. The whole package relies on one the lighting packages files. So they must be in the same catkin workspace in order to compile when performing 'catkin_make'.

Post-processing: Lighting Recommendation

Written by: Hongxiang Fu

The objective of post-processing part of this project is to identify opportunities to consume less energy while maintaining required lighting level in the facility and acceptable work amount of retrofit.

Opportunities and constraints

Lighting level constraint

The objective function of this optimization is straightforward – finding the option with lowest possible energy consumption. The first constraint of the program would be maintaining certain lighting level on the working plane. Excess lights generally exist in all kinds of buildings and facilities, although the owner usually do not prefer a lighting level exactly meeting any requirements. Extraneous lighting is usually accepted or even preferred, which renders the efforts of reducing lighting level in the space moot.

It is still plausible to set a value higher than a general metric to explore the energy saving possibility. Another opportunity lies within the situation where local or overall unacceptably high lighting level exists in the facility.

Lighting technology constraint

Luminous efficacy varies greatly among different types of bulbs or lighting technologies. Some examples are shown in the following table.

Category	Type	Overall luminous efficacy (lm/W)
Incandescent	5 – 40 – 100 W tungsten incandescent (120 V)	16.7 – 17.6 – 19.8 [7][8]
	photographic and projection lamps	35 [c20]
LED	white LED (raw, without power supply)	4.5 – 150 [9][10][11][12]
Fluorescent	9 – 32 W compact fluorescent (with ballast)	46 – 75 [7][8][14]
	T8 tube with electronic ballast	80 – 100 [13]

But industrial facilities sometimes would specify certain type of lighting technology to meet specific requirements. Incandescent light bulbs are generally of lower luminous efficacy than fluorescent light bulbs, but have a high color rendering index which is of greater importance in some situations.

A constraint of lighting technology specified by the owner must be set in the optimization.

Evenness of the brightness level

Lighting plans are usually intended to provide even and uniform brightness on the working plain. However, peak values of illuminance within the middle areas can be found in most lighting plans. These areas are of excessive lighting and can be reduced.

Study of the lighting field [6]

Fundamental concepts and symbols

Quantity name	Symbol	Unit name	Unit symbol	Notes
Luminous flux/ luminous power	Φv	lumen (= cd-sr)	lm	Luminous energy per unit time
Luminous intensity	I_v	candela (= lm/sr)	cd	Luminous power per solid angle
Luminance	L_v	candela per square meter	cd/m^2	Luminous power per unit solid angle per unit *projected* source area
Illuminance	E_v	lux (= lm/m^2)	lx	Luminous power *incident* on a surface

Properties of importance

Illuminance E_v from source point i with luminous power of Φv to incident point j can be calculated using the following formula

$$E_{\mathrm{V},j} = \sum_i \frac{\Phi_{\mathrm{V},i} \cos\theta_i}{r_{i-j}^2}$$

where θ is incident angle, r_{i-j} is the distance between the source point i and incident point j. In an coordinate system of (x,y,z), it can be expressed as

$$E_{\mathrm{V},j} = \sum_i \frac{\Phi_{\mathrm{V},i} \Delta z_{i-j}}{\left(\Delta x_{i-j}^2 + \Delta y_{i-j}^2 + \Delta z_{i-j}^2\right)^{\frac{3}{2}}}$$

This quantity is summable, and is used to express the lighting level constraint.

Recommendation program

The overall program structure is shown in Figure 29. The program first make query from 2 databases: the installation point database has information of the lighting plan of the space which is obtained from the sensor package, the bulb class library has information of the possible options of lighting technologies. After retrieving of information, the lighting plan of the space is analyzed by comparing the type and dimension values from the sensor reading to the class library. Optimization recommendations will be evaluated. The report is generated after the optimization and provide comparison information.

Figure 29: Program structure flowchart

Input

The input database structure involved in this program can be expressed in an E-R relation of Figure 30 .

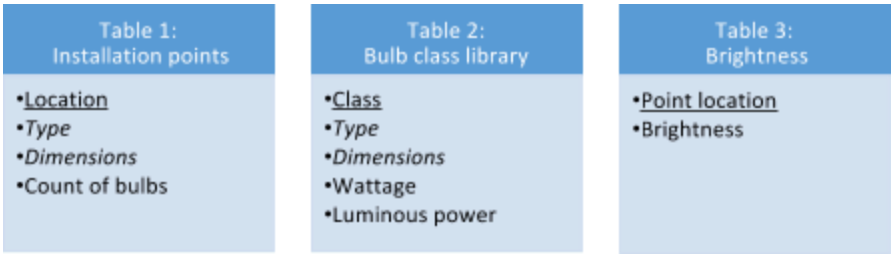


Figure 30: Input database structure

Fields with underlines are candidate keys of the table which label each entry as a unique element; *fields in italic* are foreign keys of the table which provide connection to other tables.

Lighting plan analysis

The lighting plan of the space is first identified by a physics-driven model. The model is calibrated by brightness data in the space collected by the sensor package.

1. Bulb classification

Inputs from sensors will provide values of the type, dimensions and counts of the bulb(s) installed at a certain location. The inputs are then compared to the class database to identify each bulb.

The bulb class library contains a typical bulb for the known bulb dimensions. Using tube fluorescent bulbs for example, a common value of wattage and luminous power can be found with a known pair of length and tube diameter values, incidental assumptions in order to narrow down to values widely chosen include 4100K or 5000K color temperature. Because the reading is only taken from beneath the bulb, this is not necessarily consistent with dimension values needed for classification. Tube bulbs can generally be properly processed and identified as both length and width can be precisely obtained from the sensor package. However, bulbs with shapes that only provide 1 available dimension reading do not have enough information for this process.

2. Model calibration

The calculation model established in the program is physics-driven, thus it requires calibration to reflect effective factors not considered, including shading and decay of luminous power. For simplicity, a single constant calibration coefficient is multiplied to the physical expression of illuminance calculation.

$$E_{\mathrm{V},j} = c \cdot \sum_i \frac{\Phi_{\mathrm{V},i} \Delta z_{i-j}}{\left(\Delta x_{i-j}^2 + \Delta y_{i-j}^2 + \Delta z_{i-j}^2\right)^{\frac{3}{2}}}$$

where c is the coefficient of calibration. Hence the coefficient can be found from measure brightness values $E_{\mathrm{V},j,\mathrm{mea}}$ and calculated illuminance values $E_{\mathrm{V},j,\mathrm{cal}}$.

$$c = \frac{\sum_j E_{\mathrm{V},j,\mathrm{mes}}}{\sum_j E_{\mathrm{V},j,\mathrm{cal}}}$$

This coefficient is attached in calculation henceforth.

Recommendation

There are multiple approaches can be taken to recommend possible optimization for the lighting plan aiming from different perspectives.

1. Reducing wattage

The wattage used in the space can be directly assuming the existence of excessive lighting in the space. Illuminance of the working plane is checked to maintain acceptable lighting level.

2. Delamping

Simply removing some of the bulbs in one cell is one of the easiest way to reduce energy consumption in lighting, because this requires no change of installation or any new purchase at all. Typical lighting field plans are usually accompanied with some unnecessary peaks in the lighting field and cause some degrees of over-lighting while maintaining adequate brightness at other locations on the working plain. This condition is the most suitable for the delamping process for lighting optimization.

In the program, the points on the working plane with calculated illuminance values are sorted, and the point at the 84% brightest place is used as the threshold. If a light falls within a point where the corresponding working plain point has a brightness value higher than the threshold, this light will be de-lamped. The percentage of 84% is subject to manual adjustment based on each specific condition.

3. Comparison over technologies

It is usually worth comparing the current lighting technology used with others while maintaining the same level of lighting. However, as is stated above, the capability of identifying the bulb is limited to the number of bulb dimensions exposed to the sensor package when readings are taken from beneath. For tube bulbs of fluorescent lights, a typical class can be determined with width and length inputs. However, for a typical round incandescent bulb, its "length" refers to a vertical dimension which cannot be taken from the sensor package, there is not enough information for this process. Thus this approach of optimization is not fully launched.

Report generation [15]

A Word report is generated automatically after the running of the program. In the report the illuminance field mesh plot, the illuminance contour plot and a table of bulb characteristics are compared over the original lighting plan and revised plans. The report generation program make decision on whether to generate a particular part by detecting whether the corresponding file of each part exists.

Case Study and Demonstration

A workbench area of Engineering Innovation Center on Texas A&M University campus is studied to demonstrate the characteristics of this model and optimization process. This area is

chosen because of its rectangular shape which can easily show typical features of a normally designed lighting plan.

Mind that the illuminance values are not fully calibrated, because the fluorescent bulbs are old and dim, giving the area a low level of brightness.

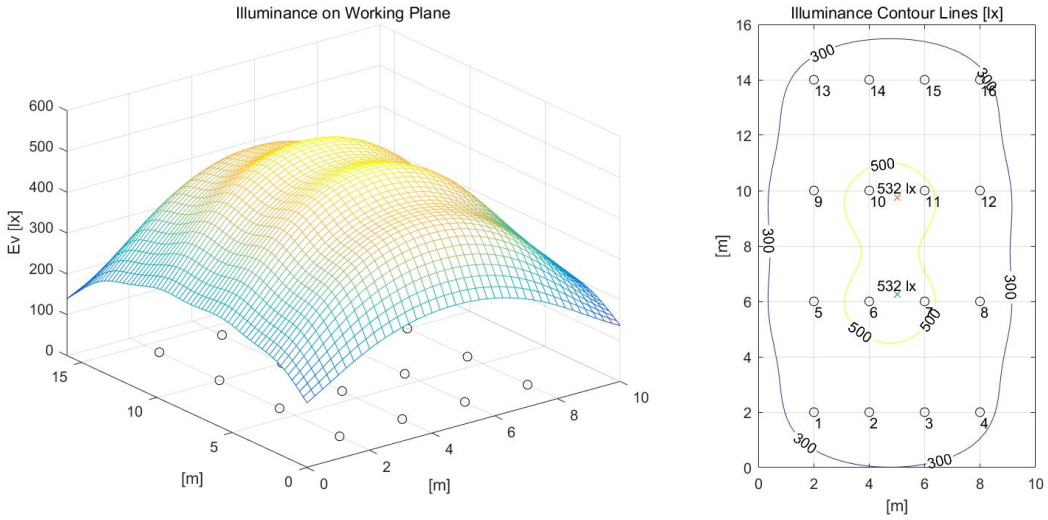


Figure 31: Illuminance Level of the Initial Lighting Plan

Reducing wattage of each bulb also reduces the overall illuminance, but does not change the shape of the plot.

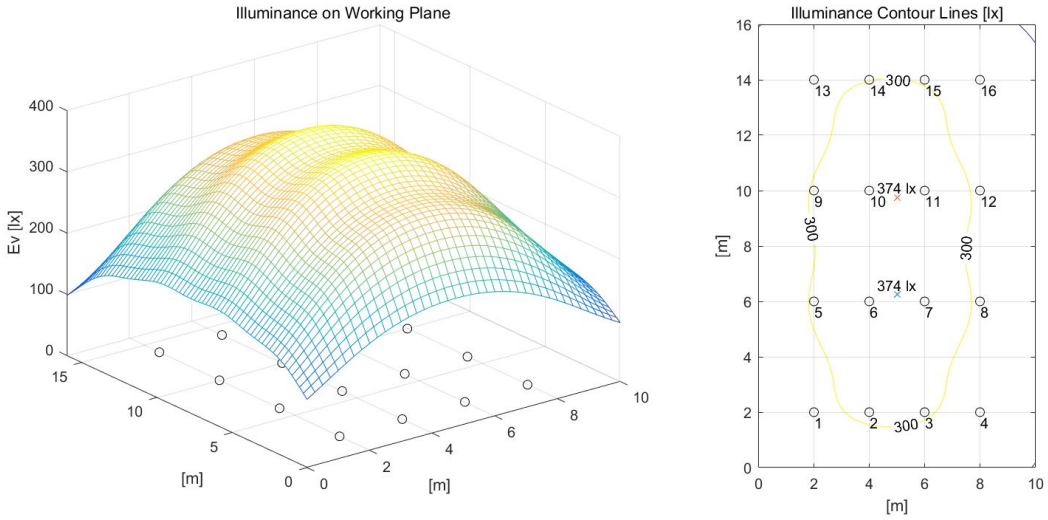


Figure 32: Illuminance Level of the Reduced Wattage Lighting Plan

The delamping process, however, effectively cuts off the peak and balances the lighting field. In this case, each light whose corresponding working plain point has illuminance greater than 482 lx has one bulb removed.

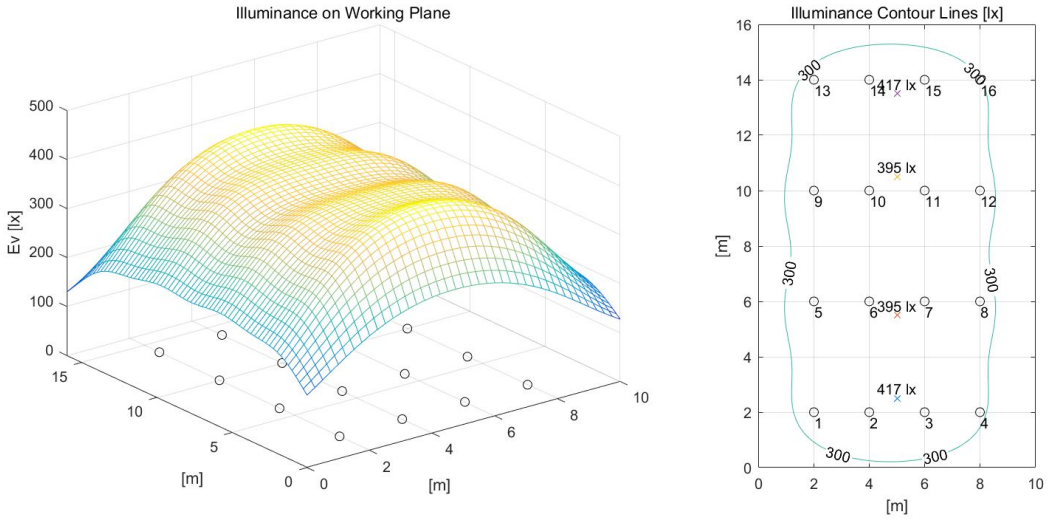


Figure 33: Illuminance Level of the De-lamped Lighting Plan

Project Timeline

The timeline for the fall semester was set by Christopher Bay and Dr. Rasmussen and is shown in Figure 34 and Figure 35.

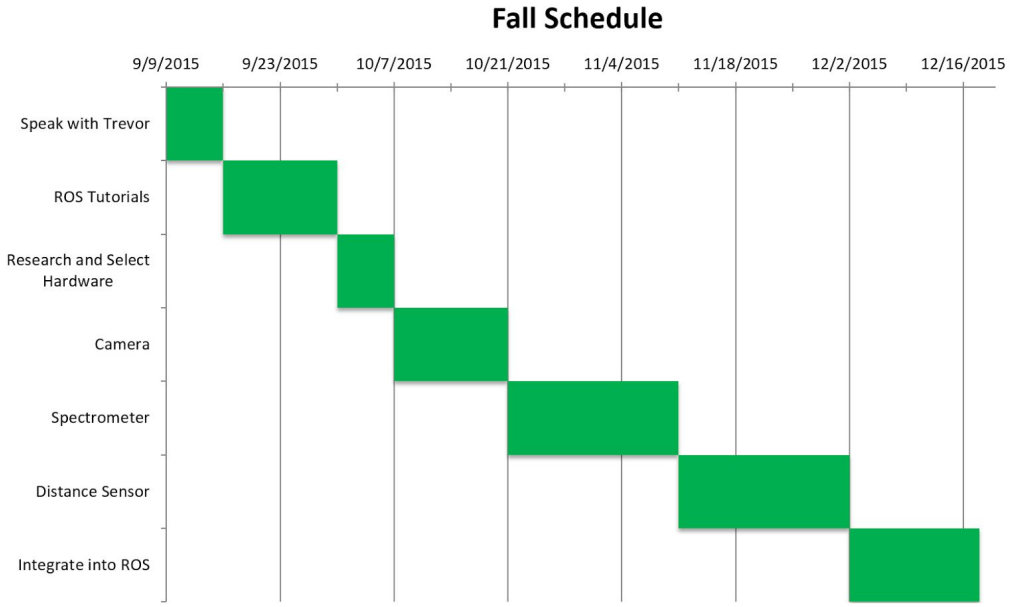


Figure 34: Fall 2015 Lighting Team Schedule

Goals for the fall semester included demonstrating ability to capture images from the camera with exposure control, record spectrometer readings from the spectrometer, and record distance using the laser distance sensor. Finally, each must be integrated into ROS for

The proposed timeline for the spring semester is shown in Figure 35, and includes a demonstration of the working product just before spring break.

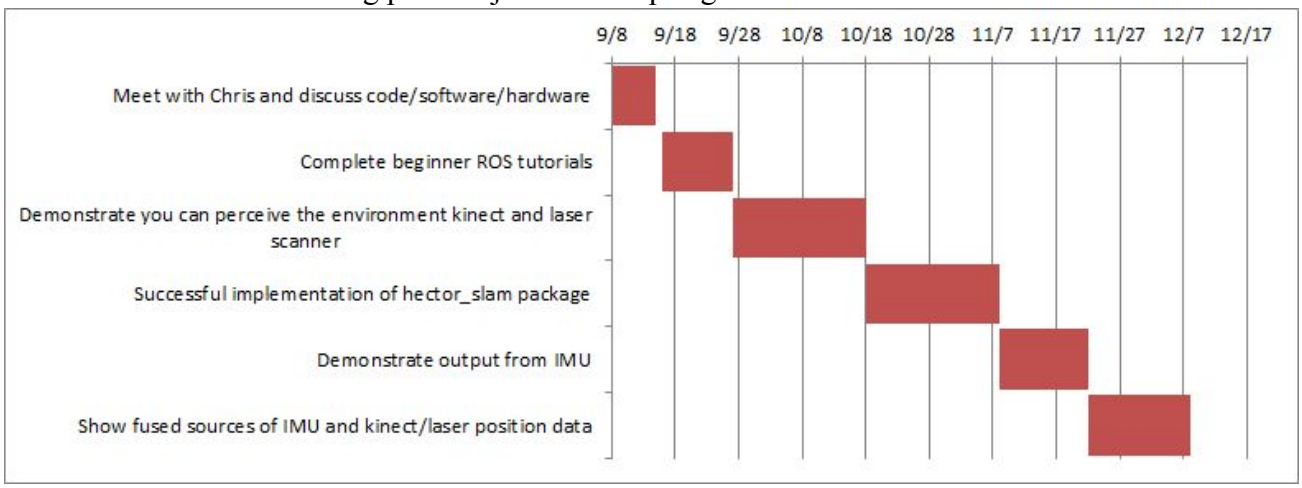


Figure 35: Fall 2015 Mapping Team Schedule

Goals for the fall semester included being able to perceive the environment using the Kinect and laser scanner, creating a 2D map with the Kinetic and laser scanner with the use of hector_slam and ROS, retrieve data from the IMU, and to be able to fuse the data from the IMU with the Kinetic and laser scanner. The mapping Team was able to accomplish most of its goals in the Fall except for being able to create a 2D map with the Kinetic and being able to integrate the IMU data with any of the scanners. This tasks were continued into the Spring as you can see by Figure 36.

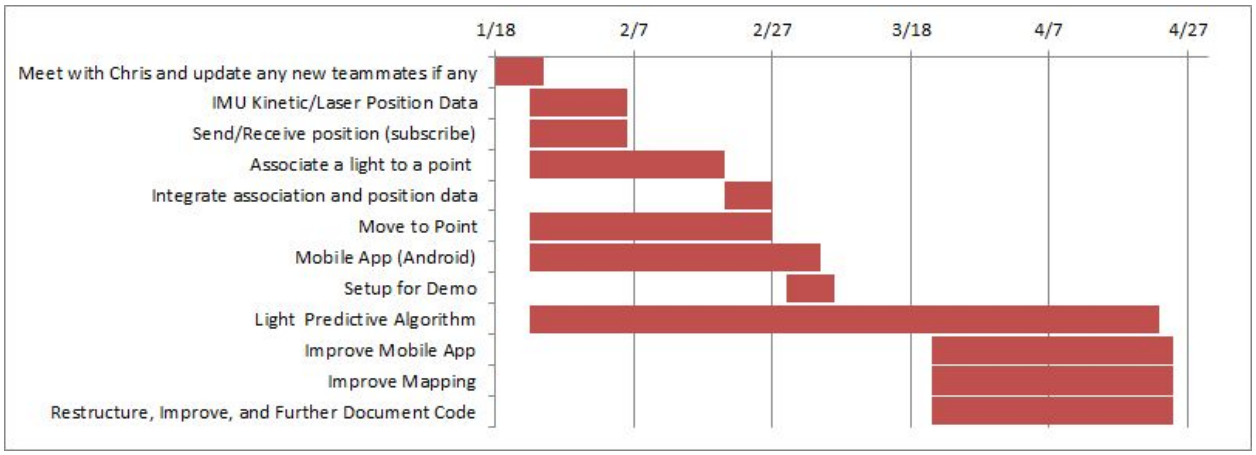


Figure 36: Proposed Spring 2016 Lighting Team Schedule

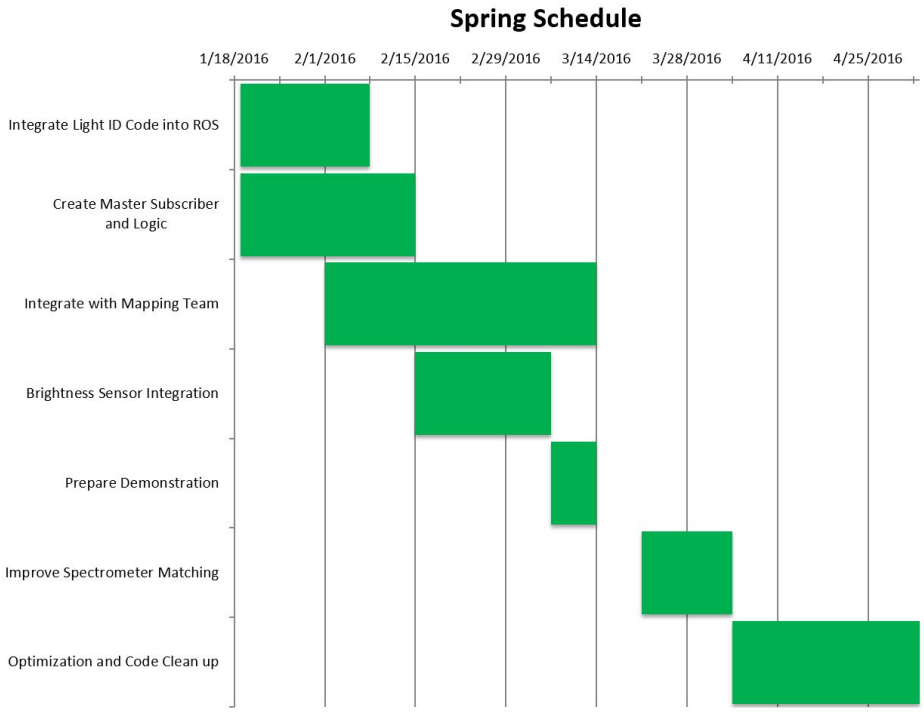


Figure 37: Proposed Spring 2016 Mapping Team Schedule

This Spring we decided on using the laser scanner as the primer sensor to create the map. We learned how to send and receive data from the lighting team and be able to show visual data so the user can see where the lights that are being found are. We worked on integrating a mobile app to be able to see the map in it as well. We didn't work on the light predictive algorithm, since we felt that, that was not something we had to worry about too much.

In the spring, the code from previous teams will be integrated into ROS, a main package combining the outputs from the camera, spectrometer, and distance sensor will be created. This package will also be the one to interpret the lighting position data from the previous team's code, and interface with the mapping team. The addition of a brightness sensor will help collect useful data for lighting audits, and will also help normalize the spectrometer readings for later comparison. As mentioned, there is a planned demonstration in mid-march, just before spring break. Following the break, spectrometer matching will be improved using the new brightness sensor data. Recorded data will be analysed and formatted for use by the post-processing program. Finally, code will be commented for use by future teams, and optimized to run as quickly and efficiently as possible.

In actuality, each section took a bit more time and this effectively dominoed to suspend the other dependent portions, but we still completed all of the sections by the end of the terminal date. One very pressing issue was the malfunctioning of the spectrometer; running a certain function took around ten minutes to complete. After a month of trying to get customer support to help, they became unresponsive until eventually suggesting to purchase a new one. Designing the subscription system was also a bit challenging, but it was completed soon thereafter.

Implementation and Results

Hardware

Camera

Written by: Ramsey Bissex

The camera worked quite well for the project. It had a fast response time and had enough definition to correctly detect lights most of the time. At the start of the project there were some problems getting it configured but after that was worked out there was no problem.

Spectrometer

Written by: Ramsey Bissex, Randy Ardywibowo

The spectrometer was capable of detecting a spectrograph of a light. There were a number of problems with it though. It would take about 10 seconds to take a reading. This is able to be controlled by settings given by the spectrometer but set to less than 10 seconds the spectrograph would have noise in the graph. The fiber optic cable was also really fragile, causing it to break at the base of the spectrometer. Because of this, an identical spectrometer was ordered, increasing the amount of money we spent.

The startup time for the spectrometer proved as a bottleneck on some configurations. This is because the startup time varied depending on which computer it was used on. This time varied from 10 seconds to 10 minutes, causing it to be unreliable during debugging situations. However, after this time passed, the spectrometer is able to function normally and reliably.

Distance Sensor

Written by: Ramsey Bissex, Randy Ardywibowo

The distance sensor had many problems through the project. There were many permissions issues, and reliability issues. Also the wires connecting two parts of the device had a tendency to come lose or even break. However, this sensor was finally programmed to work reliably with our package. This allows it to function, albeit with a low refresh rate. This slow refresh rate is caused when measuring the distance of reflective surfaces, causing the distance sensor to take more time in collecting the light emitted to infer a distance.

Several hardware issues were seen in the distance sensor. Besides the issue of the cables coming off, the distance sensor seemed to wobble as it wasn't placed in a rigid enclosure, causing noticeable errors in ceiling distance measurements. These hardware issues prompted us to create an enclosure for the distance sensor.

Brightness Sensor

Written by: Ramsey Bissex, Randy Ardywibowo

The brightness sensor worked well in the project. The hardware and software installation process was quite simple. This device came with many programming examples in many languages expediting the installation process. However, a minor issue occurred during our testing. This issue was caused by a Yoctopuce software, which caused permission conflicts for our package. To fix this, the Yoctopuce software's startup script was removed.

Case

Written by: Amir Darwesh

The case built for the project served well for securing the various sensors in place during the project demo. The enclosures were secured to the Turtlebot using zip ties which was sufficient enough so that the Turtlebot could move with speed without having the sensors jitter causing inaccurate readings during its movement. One problem encountered was that the pegs securing the brightness sensor often broke off, requiring a secondary method of securement. This is due to the fact that 3-D printing small extruded surfaces only have the layer of support it rests on. To circumvent this flaw for future designs, a better method of attachment would be to design holes that would be used to mount the brightness sensor.

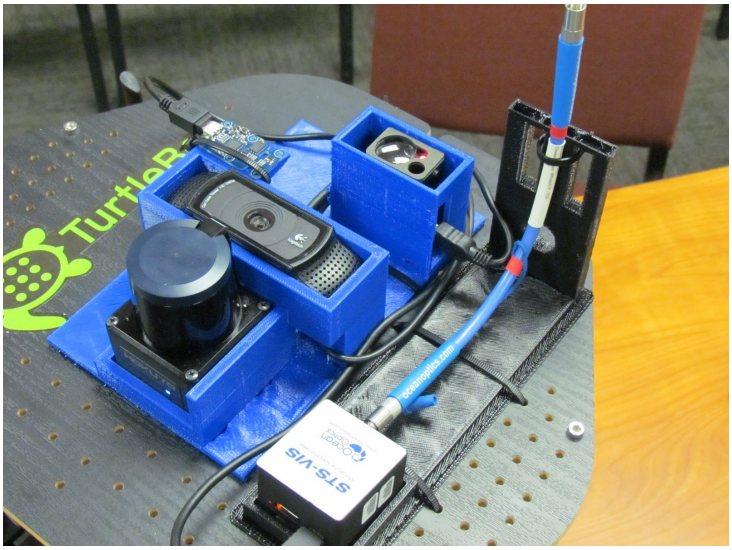


Figure 38: The final sensor configuration, attached to the turtlebot

Figure 39: Casing for Sensors[6]

Kinect

Written by: Yang Yang

The first step towards our semester goal was accomplished by being able to use the Kinect to see our environment using Rviz. Were were later able to create a three dimensional map of our environment. Since the kinect does not have built in odometry, we were only able to map with fake odometry data, causing possible exceptions during mapping, and not desirable accuracy. Also, the three dimensional map generated suffers from low-resolution. On the other hand, we have been trying to use a package called pointcloud_to_laserscan to convert the depth image into laserscan data, but this package cannot recognize our device. We found that the coverage of the point cloud and the distance was very short. So at the same time we investigated a laser scanner.

Laser Scanner

Hokuyo URG-04LX

Written by: Nicolas Botello

For the Hokuyo- URG-04LX, we were able to first perceive the environment using the hokuyo node package from ROS in Rviz. We are able to detect different points that represent obstacles that the hokuyo beams hit. After being able to retrieve that information, we then started researching how to setup the Hector SLAM (2D mapping). In order to be able to retrieve the information from the scanner we had to install a package called hokuyo_node that would run the laser scanner and retrieve its information and format it properly. We then found that it's easier if you create a package that deploys hector_slam, hokuyo_node, and Rviz. In agreement with what we found we created a package called uprobotics that contains the settings needed and packages needed when running the files in the launch file. Everytime we plugin the device we need to give permissions to the port to be able to retrieve information from is so we created a bash file that does all this for us, but in order for the script to work the usb cables must be plugged into the NUC or laptop directly to the computer without a usb hub. We found that the laser scanner needed detail in order to be able to detect where it is and if it's moving without the use of a IMU. The distance the scanner was able to pick up was four to seven meters even though the specification says it's about ten meters.

From the data we acquired from the kinect and Hokuyo we decided to keep going forth with the laser scanner, since it had more range and we were able to get a 2-D map in the time that we had.

So since we discovered some weaknesses we were able to obtain the Hokuyo UST-20LX for its better specifications and we felt that the rate at which the URG-04LX scan rate was happening was limiting us. Since when we would do some sharp turns it would lose itself. We felt that if we had a scanner that would scan at a faster rate and if we were able to add the imu we would be able to fix the problem of the laser scanner losing itself, since the hector_slam package was using a sort of point mapping stitching and estimating the movement from its last scan.

Hokuyo UST-20LX

Written by: Nicolas Botello

The Hokuyo UST-20LX has a 270° area scanning range and 25ms scan speed with a max range of 20 meters. We were able to detect areas a lot better using this sensor and detected about 10 meters.

Though this model of Hokuyo didn’t come with a usb plug to power the device since it needed 0.15A we instead created a power plugin connection that would fit to the port it came with. The board then provided some terminals to connect to the turtlebot that provided enough Amps to power it. This model also, had an ethernet cable to send the data instead of usb, but the connection was quite short so we had to build an extension. In order to utilise this model we had

to use a different ROS package called urg_node that would set up a small server and you would then have to pass the ip_adress as a param for the data to be received from.

Figure 40: The Hokuyo UST-20LX Laser scanner [18]

Inertial Measurement Unit (IMU)

Written by: Nicolas Botello

For the IMU 9dof Razor we found a package called razor_imu_9dof . In which we followed the instructions and were able to set up to retrieve information by first downloading the firmware through the use of the Arduino IDE. In which you can then be able to see the pitch, roll, and yaw, followed by then being able to run a program in ros that runs some python files to create a visual representation of the sensor and retrieving the same values.

We tried to get the imu data with the laser scanner data, but we were not successful. We then found a package called robot_localization in which it would take in different sources of data and then combine it into when. We tried to to get the data from hector_slam (laser scanner) with the data of the imu into robot_localization and back into hector_slam to update the location, but we were not able to do it. We were getting errors and were not able to get it to work with it.

Turtlebot

Written by: Nicolas Botello

We first attempt to use the turtlebot without hector_slam or adding any outside sensors. The turtlebot system was able to create a map using gmapping and perform an automated navigation within this map, using AMLC algorithm. The system (the so called turtlebot) consists of a rounded robot under shelves on which the Intel NUC and its power supply circuit lies upon. A

Microsoft Kinect is mounted in the structure as well. The Kinect can feed itself from Kobuki's battery by adapting its power supply cord.

Kobuki comes with encoders built in and the navigation algorithms use them to form a more accurate map, with the use of Kinect. Having the Intel NUC remotely accessed - on top of turtlebot - it was possible to create a very accurate map of an approximately 15 by 6 feet room. Kinect gives a good estimation of its environment within the section where it's vision can reach; odometry from the Kobuki helps keeping track of position and variations of it. Rotating turtlebot is often necessary to obtain a full map but the wheeled robot is not slow and it doesn't take much to calculate the distances from its surroundings forming a map. After a room's map is obtained the robot uses an estimation of its position and orientation given by the user to be able to drive itself anywhere within the room, with any orientation given by the user.

Router

After our careful consideration to not use the kinect and instead use the laser scanner we created a casing in which we the laser scanner would sit and other lighting sensors. We then had the Intel NUC onboard as well like in the scenario done before and had a usb hub to plug in the different sensors and a battery to power the NUC. The turtlebot was used to move and map the environment by executing the turtlebot ros package which is launched by the slam.launch file that includes the turtle_bot.launch file that launches all needed nodes to be able to drive the turtlebot. Though the turtlebot was teleop not autonomous we had to drive it using the keyboard of a laptop that would generally ssh into the NUC onboard of the turtlebot. The NUC onboard of the turtlebot has a script that would run when turn on and would sign in into the schools internet and would send an email to the emails on the script with the IP of the NUC,so we could then ssh in.

The reason the turtlebot was used was as a prototype instead of using a UAV, so we are able to perform faster testing and keep the same axis. In order to be able to port it into a handheld/UAV we need to integrate the IMU sensor, so we are able to detect tilt and adapt the placement of the map .

Tablet

Written by: Kevin Patel

To integrate the tablet with the system we were required to use a open source applications for android tablets. Since the tango is brand new to the market there was not any software that we could use to direct map with the built in cameras. While using the built in camera would have been ideal, we were required to outside applications to do . We had two methods to do this the first being RViz for android , and the second being Romo Remocon.

RViz for Android

RViz for android was our original solution to making our package a handheld device. RViz for android is a application written for android by Adam Zimmerman at Willow Garage. This application is to be downloaded from Zimmerman's bitbucket repository. This application uses

OpenGL to render all graphics and textures. While this application was meant to work with RViz it was not the solution as did not have any method to control the turtlebot. This application as well could not read any outside data.

Rocon Remocon

We decided to use Rocon Remocon to control the turtlebot and as well visualize the mapping in real time. This application can be found on the google play store and is to be used on android tablets. This application was created by Yujin robotics the company who created the turtlebot. Using this app we were able to easily control the turtlebot from the google tango. To get map on the tablet all we had to simply do was push the to tablet and the map would render in real time.This application is flawed in the fact it requires the turtlebot to be connected to same network as the the computer running the ros master. The map generated on to the tablet is done using gmapping. The application pulls the data straight from the master and then uses it own algorithm in rendering the map. The major flaw in this method would be that map has size restrictions places on it. After rendering a certain size tablet is no longer able to render to tablet screen , but RViz is still collecting data and rendering the map on the ROS master. This size is mainly due to the tablets' limitation it simply does not have the processing power nor the memory size to render very large map using it own gmapping algorithm.

RViz

Written by: Yang Yang

During the assessment, all results are directly displayed in RViz, a 3D visualization tool of ROS, including the current map, and lights found. Note that the lights displayed are interactive markers, and when users press any of them, it would send a request (via ROS server-client package) to the lighting team to let the web-camera take a picture of the light. The implementation is mainly in button.cpp, which subscribes to **\lighting\light_location_package**, messages are transmitted in form of **light_location_package\LightData**, which includes the light's ID and world position. A variable, max, is used to keep track of the largest ID. When message is received, if its ID has not been seen before(larger than max), then create a new interactive marker, add it to interactive marker server, and increment max by one. Else if ID is smaller than max, then get the corresponding marker from interactive marker server, and apply new position to it, then push the marker back into the server.

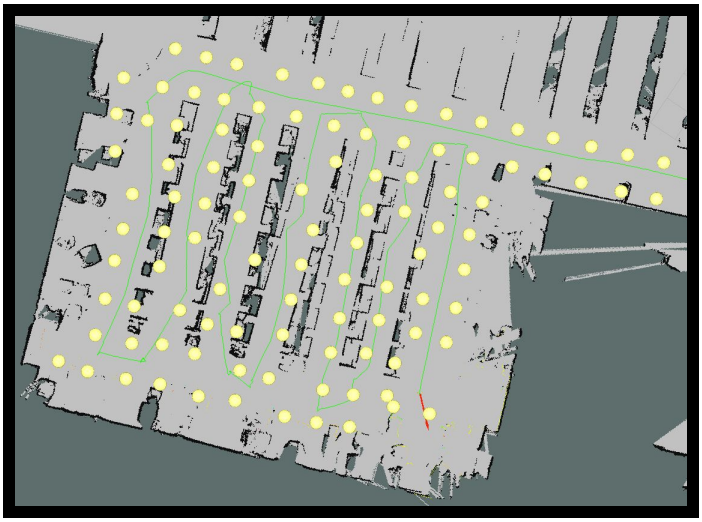


Figure 41: Library 6th floor section map

Demonstration and Testing

Written by: Randy Ardywibowo

To test our product, we conducted two tests in two separate places, the 6th floor of Evans Library and the 3rd floor of the Mechanical Engineering Building (MEOB). During both of our tests, the robot was able to map the building and locate the lights in it reliably. However, because the Inertial Measurement Unit (IMU) was not integrated into our package, the building mapping will have errors whenever an object moves within the view of the laser distance sensor. This will cause deviations in the generated map, causing major errors in the map as existing inaccuracies to propagate to map sections generated in the future.

On the other hand, assuming accurate position data from the mapping program, the light detection algorithm is very reliable. With more than 95% of lights detected and less than 5% false positives remaining. The undetected lights are caused by several situations. Firstly, when lights are obstructed by objects, it is impossible for the light detection algorithm to find the lights as the camera's view is obstructed by the object. Secondly, when lights are within the location threshold set on the light location algorithm, the program will simply ignore the light as it is treated to be the same as an existing light.

False positives are caused by several conditions. Firstly, in some occasions network issues cause latencies in data transfer, causing the same lights to be counted twice as the distance threshold between lights are never taken into effect. Second, when the distance sensor measuring distance to the ceiling is obstructed by some object, the program will detect new lights in a different plane.

Figure X illustrates these problems. Although a good map of the building was generated with all lights detected, some boundary cases where the package failed did occur. On the top right of this figure, the mapping algorithm deviated slightly from the actual building plan. This can be

seen from the deviation angle from the actual building plan. This deviation angle, although small, can cause major issues as the error propagates to futures map sections generated in the building. Secondly, the program's light location threshold caused several false negatives to occur. This is because some lights are within the threshold set by the program. The threshold was set to 2 meters in this demonstration and has since been lowered to 0.5 meters, which has been tested to be more reliable.

Figure 42: Generated Map of 3rd Floor of the MEOB Building

Budget

The overall budget for the project is shown in Table 1.

Item	Cost
Omnidriver	$500.00
Intel NUC	$600.00
Logitech C920 Webcam	$100.00
Ocean Optics STS-VIS	$1459.00
Porcupine Labs LR4	$149.00
Yocto-Light-V3	$25.00
Kinect for Xbox	$25.00
Hokuyo URG-04LX	$1115.00
Hokuyo UST-20LX	$3000.00
Razor IMU 9dof	$75.00
Router	$50.00
ABS filament for case	$20.00
Total	$7098.00

Table 1: Overall Team Budget

Future Directions

Written by: Garrison Neel, Randy Ardywibowo

To improve the current prototype, there is much that could be done. In the area of light detection, stereo cameras could be implemented in order to gather distance data for each pixel. Per-pixel distance data would eliminate the need to assume a level ceiling plane, and would allow the system to survey areas with varying ceiling height, or with suspended lights, both of which are common in industrial settings. 3D data from stereo cameras could also be used to eliminate reflections and false positives by recognizing when a detected light is an unexpected 3D shape.

Light thresholding could be improved with the integration of brightness sensor data. Correlating brightness value with the light detection threshold will allow for more robust adaptive thresholding that can be used in any environment. Cursory studies were performed and revealed that this method is powerful but will need some effort to properly configure.

To improve the reliability of the light detection algorithm, a statistical position thresholding method should be used. This method should take into account when false positives and false negatives are detected and adjust the statistical likelihood of a light existing accordingly. This would improve the light detection and location when objects block the camera's view of the light.

In the area of mapping, there is a big problem to solve, which is integrating the IMU sensor into the SLAM, so we are able to get a more precise position of our current position. After the IMU sensor is able to be integrated with the data from the laser scanner we will be able to tell if the user is tilting and rotation, so that it isn't able to lose itself. After the integration with the IMU is done, it will be a lot easier to perform autonomous movement. This can be done by using an Extended Kalman Filter (EKF) algorithm to continuously compare the robot's pose as stated by the IMU and the Hector SLAM algorithm.

Reducing the size and weight of the package is essential to its versatility. Currently, the system is light enough to be human portable, but will require some auxiliary supporting structure, such as a backpack, to contain the NUC and batteries. Small form-factor computers like Intel's compute stick or more capable systems in the future could be used to reduce power requirements and therefore weight.

Conclusion

Written by: Garrison Neel

Over the past year, we have achieved many of our goals. All sensors communicate through ROS, lights are detected and placed correctly within the map, and lights are correctly identified by their bulb dimensions and emissions spectrum. For mapping, laser scanner data was used with Hector SLAM to create a 2D map. Together in the prototype, systems function together in real

time to capture data. The post processing program generates an automated report with suggested changes, generated from data captured by the package.

This project has very promising future prospects. With a functioning prototype, the foundation of the system has been laid. Future teams can build upon existing work to make more rapid improvements toward enhancing the robustness of the system as well as its packaging.

References

[1] U.S Department of Energy, “Commissioning for Federal Facilities,” [Online]. Available: www1.eere.energy.gov/femp/pdfs/commissioning_fed_facilities.pdf. [Accessed 10 December 2015].

[2] "Logitech C920 Camera." Logitech. 2 July 2013. Web. 3 May 2016. https://secure.logitech.com/en-us/product/hd-pro-webcam-c920

[3] "STS VIS Microspectrometer." Ocean Optics. Ocean Optics, 4 July 2014. Web. 3 May 2016. http://oceanoptics.com/product/sts-vis-microspectrometer/

[4] "LR4 Distance Sensor." Porcupine Labs. Porcupine Labs, 21 June 2014. Web. 3 May 2016. http://www.porcupinelabs.com/lr4/

[5] "Yocto Light V3." Yoctopuce. Yoctopuce, 24 Mar. 2012. Web. 3 May 2016. http://www.yoctopuce.com/EN/products/usb-environmental-sensors/yocto-light-v3

[6] Simons, R., & Bean, A. (2001). Lighting engineering applied calculations. Oxford: Architectural Press.

[7] Philips Product Catalog (German)

[8] "Osram halogen" (PDF). osram.de (in German). Archived from the original (PDF) on November 7, 2007. Retrieved 2008-01-28.

[9] Klipstein, Donald L. (1996). "The Great Internet Light Bulb Book, Part I". Retrieved 2006-04-16.

[10] "White LED Offers Broad Temp Range And Color Yield". Electronicdesign. 2001-04-02. Retrieved 2013-05-16.

[11] "Nichia NSPWR70CSS-K1 specifications" (PDF). Nichia Corp. Retrieved 2013-05-16.

[12] Klipstein, Donald L. "The Brightest and Most Efficient LEDs and where to get them". Don Klipstein's Web Site. Retrieved 2008-01-15.

[13] "Cree XLamp XP-G LEDs Data Sheet" (PDF).

[14] Federal Energy Management Program (December 2000). "How to buy an energy-efficient fluorescent tube lamp". U.S. Department of Energy.

[15] Laurent Vaylet. Wordreport - File Exchange. http://www.mathworks.com/matlabcentral/fileexchange/17953-wordreport

[16] "XBOX 360 Kinect." XBOX 360 Kinect. Microsoft, 20 Sept. 2010. Web. 3 May 2016. http://compass.xbox.com/

[17] "Hokuyo URG 4LX Laser Rangefinder." Robot Shop. Robot Shot, 11 June 2015. Web. 3 May 2016. http://www.robotshop.com/en/hokuyo-urg-04lx-laser-rangefinder.html

[18] "Hokuyo UST 20LX Laser Rangefinder." Robot Shop. Robot Shot, 11 June 2015. Web. 3 May 2016. http://www.robotshop.com/en/hokuyo-ust-20lx-scanning-laser-rangefinder.html

[19] "9 Degrees of Freedom Razor IMU." Sparkfun. Sparkfun, 29 Aug. 2014. Web. 3 May 2016. https://www.sparkfun.com/products/10736

[20] "Turtlebot." Turtlebot. Turtlebot, 23 June 2011. Web. 3 May 2016. http://www.turtlebot.com/

[21] "Project Tango." Google Project Tango. Google, 25 Nov. 2012. Web. 3 May 2016. https://developers.google.com/project-tango/images/index/tango-hero.png

[22] "TP-LINK - Wireless N750 Wireless Dual-Band Gigabit Router - Black." Amazon. Amazon, 3 Feb. 2013. Web. 3 May 2016. http://ecx.images-amazon.com/images/I/61edmMVCwmL._SL1280_.jpg

[23] Richard Hartley and Andrew Zisserman (2003). *Multiple View Geometry in computer vision*. Cambridge University Press. ISBN 0-521-54051-8.

[24] NASA Mission Planning and Analysis Division. "Euler Angles, Quaternions, and Transformation Matrices". NASA. Retrieved 12 January 2013.

[25] "Intel NUC." Intel NUC. Intel, 26 Aug. 2014. Web. 3 May 2016. http://www.intel.com/content/www/us/en/nuc/overview.html

Appendix

The following is a sample report of post-processing.

The following MATLAB codes are used to as the main body of post-processing.

```
clear, clc, close all
% Order of files: PPP, PPP_DL, PPP_Report
if exist('figDelamp.fig','file')
    delete('figDelamp.fig');
end
if exist('figOriginal.fig','file')
    delete('figOriginal.fig');
end
if exist('figReduced.fig','file')
    delete('figReduced.fig');
end

%% 1. Input
% Table of lights (3NF)
% 2 - x; 3 - y; 4 - z (height from WP); 5 - Type
% 6 - counts; 7 - length (ft); 8 - width (in)
[Tmp_1,Tmp_2,Tmp_3]=xlsread('Dummy_Lights.xlsx','Points');
cellPoints=Tmp_3(2:length(Tmp_3),:);
% length of the list
lP=size(cellPoints,1);
P_location=cell2mat(cellPoints(:,2:4)); % Location of Point
P_type=cell2mat(cellPoints(:,5)); % Type of Points
P_count=cell2mat(cellPoints(:,6)); % Counts of bulbs
P_dim=cell2mat(cellPoints(:,7:8)); % Dimensions of bulbs
clear Tmp_1 Tmp_2 Tmp_3
%}

% Table of bulb classes (3NF)
% 1 - class #; 2 - type; 3 - wattage; 4 - luminous power (lm);
% 5 - class; 6 - diameter (in); 7 - length (in)
[Tmp_1,Tmp_2,Tmp_3]=xlsread('Dummy_Lights.xlsx','Class');
cellClass=Tmp_3(2:length(Tmp_3),:);
% length of the list
lC=size(cellClass,1);
C_type=cell2mat(cellClass(:,2));
C_class=cellClass(:,5);
C_Pwr=cell2mat(cellClass(:,3));
C_PHIv=cell2mat(cellClass(:,4)); %lm4
C_dim=[cell2mat(cellClass(:,7))/12 cell2mat(cellClass(:,6))];
clear Tmp_1 Tmp_2 Tmp_3

% Table of brightness in space (3NF)
```

```
% 1 - x; 2 - y; 3 - z; 7 - illuminance (lx)
[Tmp_1,Tmp_2,Tmp_3]=xlsread('Dummy_Lights.xlsx','Brightness');
cellBright=Tmp_3(2:size(Tmp_3,1),:);
% length of the list
lB=size(cellBright,1);
B_location=cell2mat(cellBright(:,2:4)); % Location of lx reading
B_Ev=cell2mat(cellBright(:,7)); % Illuminance in lx
clear Tmp_1 Tmp_2 Tmp_3

%% 2. Classification
P_Class=zeros(lP,1);
for kp=1:lP
    minCompr=inf;
    for kc=1:lC
        if (P_type(kp)==C_type(kc))
            Compr=(P_dim(kp,1)-C_dim(kc,1))^2+(P_dim(kp,2)-C_dim(kc,2))^2;
            if (Compr<minCompr)
                minCompr=Compr;
                P_Class(kp)=kc;
            end
        end
    end
end

P_PHIv=transf(C_PHIv,P_Class);
P_Pwr=transf(C_Pwr,P_Class);
P_ttPwr=sum(P_Pwr.*P_count);
Dmn=[10 16]; % dimensions of the space (2-D)
% Walls
Wls=[];
%Wls=[3 7; 5 7; 7 7];
%Wls=[5 0; 5 10];
%Wls=[5 0; 5 9; 8 9];
%Wls=[4.5 7; 5.5 7; 5.5 9; 4.5 9; 4.5 7];
%Wls=[5 5; 7 5; 7 11; 5 11; 5 5];
clear kp kc Compr minCompr

%% 3. Search & Calibration
W_Stp=0.25; % step of space accuracy
coe=0.2; % calibration coefficient
%coe=Calib(P_location,P_PHIv,P_count,B_location,B_Ev);
W_Ev=coe*EvField(P_location,P_PHIv,P_count,Dmn,W_Stp,Wls);
[W_L_min,W_Ev_min]=minPoint(W_Ev);
lW=length(W_Ev_min);
% W_L_min - the list of locations (2-D) of min Ev values on WP
% W_Ev_min - the list of values of min Ev on WP corresponding to locations

clear k kp kf px py figOriginal cc ch

%% 4. Optimisation / Reduction
% Pnts are changed step by step and WP is checked accordingly.
% creating working vars for Pnts and WP
P_Class_wk=P_Class;
P_Class_history=[];%P_Class_wk;
```

```
P_PHIv_wk=P_PHIv;
P_PHIv_history=[];%P_PHIv_wk;
P_Pwr_wk=P_Pwr;
P_Pwr_history=[];%P_Pwr_wk;
P_ttPwr_history=[];
W_Ev_wk=W_Ev;
%W_Ev_history(:,:,1)=[];%W_Ev_wk;
W_Ev_min_wk=W_Ev_min;
W_Ev_min_history=[];%W_Ev_min_wk;
kh=0;
flag=C_typeCheck(C_type,P_Class_wk,0);% flag on if it's the end of the list
while flag==0
    kh=kh+1;
    P_Class_wk=P_Class_wk+1;
    P_PHIv_wk=transf(C_PHIv,P_Class_wk);
    P_Pwr_wk=transf(C_Pwr,P_Class_wk);
    %kw=kw+1;
    P_Class_history=[P_Class_history P_Class_wk];
    P_PHIv_history=[P_PHIv_history P_PHIv_wk];
    P_Pwr_history=[P_Pwr_history P_Pwr_wk];
    P_ttPwr_history=[P_ttPwr_history sum(P_Pwr_wk.*P_count)];
    W_Ev_wk=coe*EvField(P_location,P_PHIv_wk,P_count,Dmn,W_Stp,Wls);
    [W_L_min_wk,W_Ev_min_wk]=minPoint(W_Ev_wk);
    W_Ev_history(:,:,kh)=W_Ev_wk;
    W_Ev_min_history=[W_Ev_min_history W_Ev_min_wk];
    flag=C_typeCheck(C_type,P_Class_wk,flag);
    flag=(max(max(W_Ev_wk))<350);
end
lh=size(P_Class_history,2); % length of the history matrices.
% Record of reduced lighting plan history.
save('matReduced.mat','P_Class_history');
save('matReduced.mat','P_PHIv_history','-append');
save('matReduced.mat','P_Pwr_history','-append');
clear kh kw flag P_Class_wk P_PHIv_wk P_Pwr_wk W_Ev_wk W_L_min_wk W_Ev_min_wk

%% 5. Report Data Preparation
% This part prepares the figures and tables to be put into the report.
% The report generation codes are moved to a standalone file.

% ORIGINAL PLAN
figOriginal=plotBoth(W_Ev,P_location,Dmn,W_Stp,Wls);
tbOriginal=cellTable(P_Class,P_count,cellClass);
% Record of original lighting plan.
save('matOriginal.mat','P_Class');
save('matOriginal.mat','P_PHIv','-append');
save('matOriginal.mat','P_Pwr','-append');
savefig(figOriginal,'figOriginal.fig');
close(figOriginal);
%
% REDUCED PLANS
for kf=1:lh
    figReduced(kf)=plotBoth(W_Ev_history(:,:,kf),P_location,Dmn,W_Stp,Wls);
    tbReduced{kf}=cellTable(P_Class_history(:,kf),P_count,cellClass);
end
```

```
savefig(figReduced,'figReduced.fig');
close(figReduced);
close all
clear kf
```

The following MATLAB codes are used to as the delamping process.

```
%% Delamping programme
% PPP must be run prior to this.

% Input: delamp the Points in the following array
%Delamp=[6 7 10 11];
Ev_tmp=sort(reshape(W_Ev,[],1));
% Setting the threshold above which the bulb at the corr. pnt be delamped.
W_Ev_pk=Ev_tmp(round(length(Ev_tmp)*0.84));
% 84% is chosen to be the luck # b/o 68-95-99.7 rule, 50+68/2=84.
Delamp=[];
for kp=1:lP
    if W_Ev(P_location(kp,1)/W_Stp,P_location(kp,2)/W_Stp)>W_Ev_pk
        Delamp=[Delamp kp];
    end
end

P_count_dl=P_count;
for kd=1:length(Delamp)
    if P_count_dl~=0
        P_count_dl(Delamp(kd))=P_count_dl(Delamp(kd))-1;
    end
end
W_Ev_dl=coe*EvField(P_location,P_PHIv,P_count_dl,Dmn,W_Stp,Wls);
P_ttPwr_dl=sum(P_Pwr.*P_count_dl);
% DELAMPED

figDelamp=plotBoth(W_Ev_dl,P_location,Dmn,W_Stp,Wls);
tbDelamp=cellTable(P_Class,P_count_dl,cellClass);
savefig(figDelamp,'figDelamp.fig');
close all
clear Ev_tmp kp kd
```

The following MATLAB codes are used to generate the report.

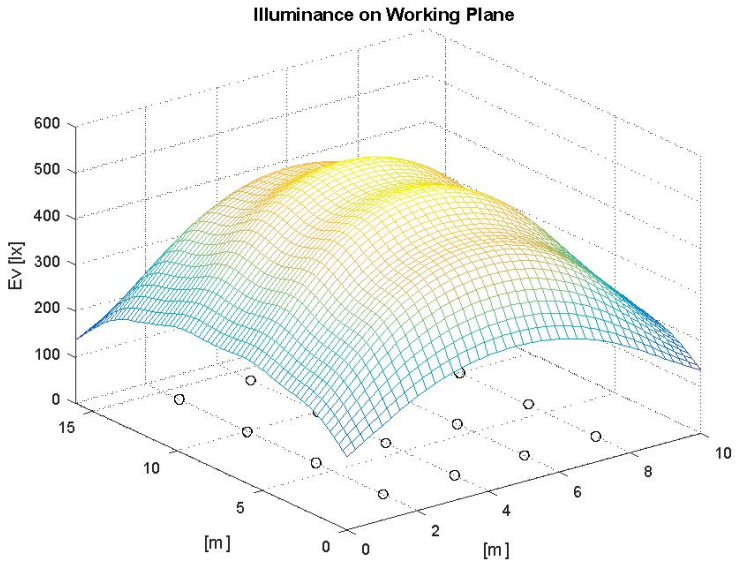
```
%% This standalone programme is used to generate the report
% PPP must be run prior to this.

if exist('Report.doc','file')
    delete('Report.doc'); % Delete the old report file
end
reportFilename = fullfile(pwd,'Report.doc');
wr = wordreport(reportFilename);

% Original
if exist('figOriginal.fig','file')
```

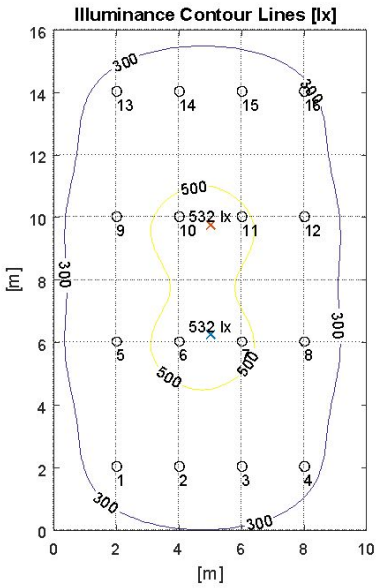
```
      wr.setstyle('Heading 1');
        wr.addtext('Original Lighting Plan', [1 1]); % line break before and
after text
      % Plot original plan
      wr.setstyle('Heading 2');
        wr.addtext('Overall Working Plain Illuminance', [0 1]); % line break
after text
      figure=openfig('figOriginal.fig');
      %figure=hMultiFig('figOriginal.fig',1);
      wr.addfigure();
      % Table of original lamps
      wr.setstyle('Heading 2');
      wr.addtext('Original bulbs', [0 1]); % line break after text
      wr.setstyle('Normal');
        wr.addtext(['Total wattage: ' num2str(P_ttPwr) ' W'], [0 1]); % line
break after text
      dataCell = tbOriginal;
      [nbRows, nbCols] = size(dataCell);
      wr.addtable(nbRows, nbCols, dataCell, [0 1]); % line break before table
      close all
  end

  % Reduced
  if exist('figReduced.fig','file')
      wr.setstyle('Heading 1');
       wr.addtext('Reduced Lighting Plan', [1 1]); % line break before and after
text
      for kh=1:lh
          % Plot reduced plan
          wr.setstyle('Heading 2');
             wr.addtext(['Overall Working Plain Illuminance ' num2str(kh)], [0
1]); % line break after text
          figure=hMultiFig('figReduced.fig',kh);
          wr.addfigure();
          % Table of reduced bulbs
          wr.setstyle('Heading 2');
          wr.addtext('Reduced bulbs', [0 1]); % line break after text
          wr.setstyle('Normal');
              wr.addtext(['Total wattage: ' num2str(P_ttPwr_history(kh)) ' W,
saving: '...
              num2str(P_ttPwr - P_ttPwr_history(kh)) ' W/' ...
                num2str((P_ttPwr - P_ttPwr_history(kh))/P_ttPwr*100,2)'%'], [0
1]); % line break after text
          dataCell = tbReduced{kh};
          [nbRows, nbCols] = size(dataCell);
            wr.addtable(nbRows, nbCols, dataCell, [0 1]); % line break before
table
          wr.addpagebreak();
      end
  end

  % Delamped
  if exist('figDelamp.fig','file')
      wr.setstyle('Heading 1');
```

```
        wr.addtext('Delamped Lighting Plan', [1 1]); % line break before and
after text
      % Plot reduced plan
      wr.setstyle('Heading 2');
        wr.addtext('Overall Working Plain Illuminance', [0 1]); % line break
after text
      figure=openfig('figDelamp.fig');
      wr.addfigure();
      % Table of delamped bulbs
      wr.setstyle('Heading 2');
      wr.addtext('Delamped bulbs', [0 1]); % line break after text
      wr.setstyle('Normal');
       wr.addtext(['Total wattage: ' num2str(P_ttPwr_history(kh)) ' W, saving:
'...
          num2str(P_ttPwr - P_ttPwr_dl) ' W/' ...
            num2str((P_ttPwr - P_ttPwr_dl)/P_ttPwr*100,2) '%'], [0 1]); % line
break after text
      dataCell = tbDelamp;
      [nbRows, nbCols] = size(dataCell);
      wr.addtable(nbRows, nbCols, dataCell, [1 1]); % line break before table
      wr.addpagebreak();
      close all
  end

  wr.close();
  clear kf kh nbCols nbRows dataCell
```

Original Lighting Plan

Overall Working Plain Illuminance





Original bulbs

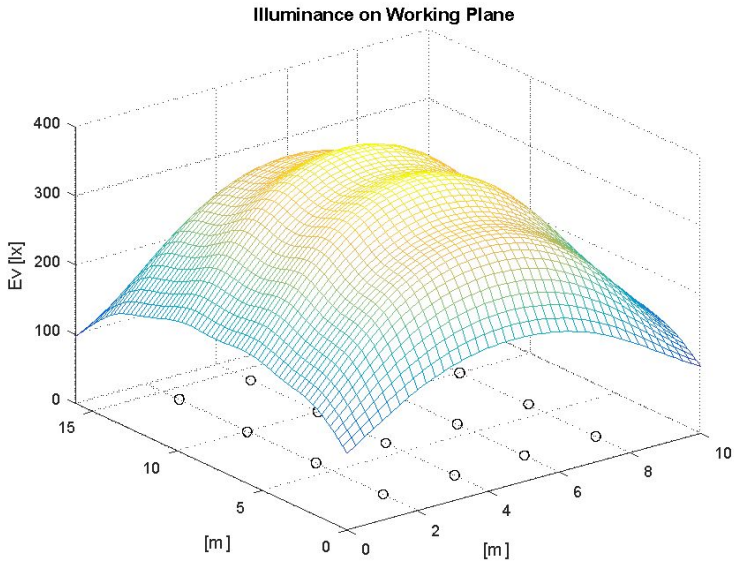
Total wattage: 1024 W

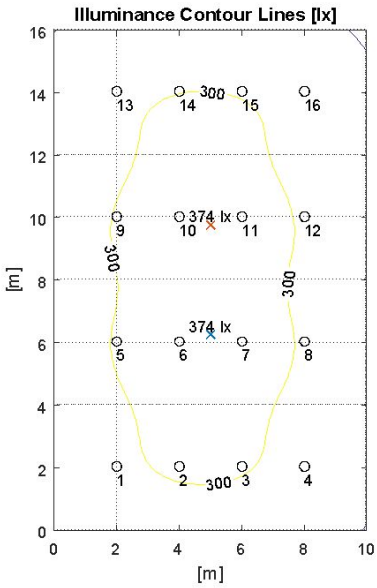
Light ID	Type	Class	Number of bulbs	Wattage each	Lumen(max) each
1	Fluorescent	W32T8	2	32	3200
2	Fluorescent	W32T8	2	32	3200
3	Fluorescent	W32T8	2	32	3200
4	Fluorescent	W32T8	2	32	3200
5	Fluorescent	W32T8	2	32	3200
6	Fluorescent	W32T8	2	32	3200
7	Fluorescent	W32T8	2	32	3200
8	Fluorescent	W32T8	2	32	3200
9	Fluorescent	W32T8	2	32	3200
10	Fluorescent	W32T8	2	32	3200
11	Fluorescent	W32T8	2	32	3200
12	Fluorescent	W32T8	2	32	3200
13	Fluorescent	W32T8	2	32	3200
14	Fluorescent	W32T8	2	32	3200
15	Fluorescent	W32T8	2	32	3200
16	Fluorescent	W32T8	2	32	3200

Table 2: Original bulbs

Reduced Lighting Plan

Overall Working Plain Illuminance 1





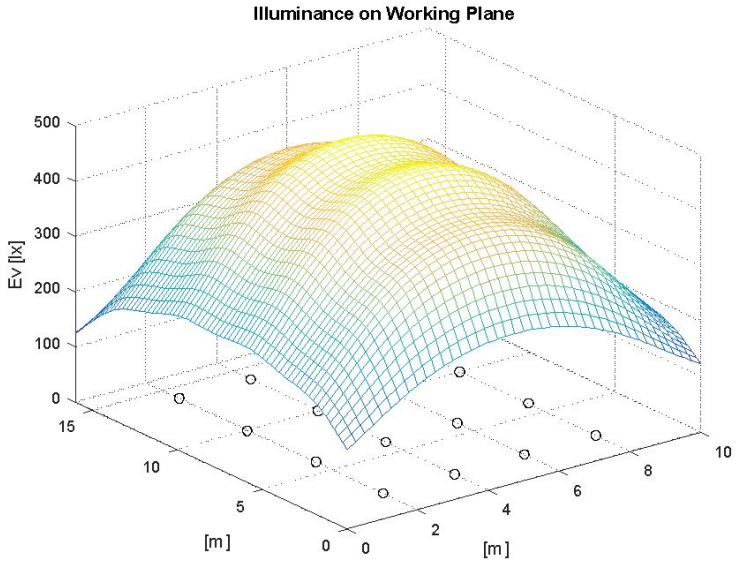
Reduced bulbs

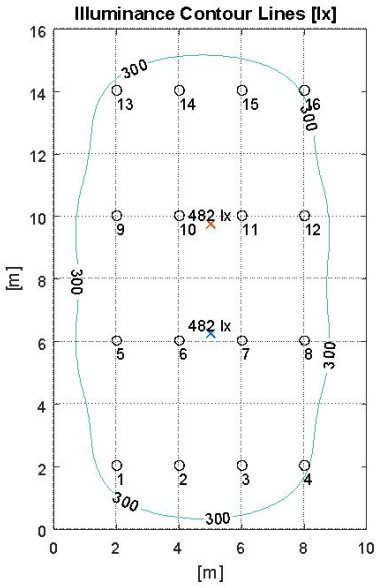
Total wattage: 960 W, saving: 64 W/6.3%

Light ID	Type	Class	Number of bulbs	Wattage each	Lumen(max) each
1	Fluorescent	W30T12	2	30	2250
2	Fluorescent	W30T12	2	30	2250
3	Fluorescent	W30T12	2	30	2250
4	Fluorescent	W30T12	2	30	2250
5	Fluorescent	W30T12	2	30	2250
6	Fluorescent	W30T12	2	30	2250
7	Fluorescent	W30T12	2	30	2250
8	Fluorescent	W30T12	2	30	2250
9	Fluorescent	W30T12	2	30	2250
10	Fluorescent	W30T12	2	30	2250
11	Fluorescent	W30T12	2	30	2250
12	Fluorescent	W30T12	2	30	2250
13	Fluorescent	W30T12	2	30	2250
14	Fluorescent	W30T12	2	30	2250
15	Fluorescent	W30T12	2	30	2250
16	Fluorescent	W30T12	2	30	2250

Table 3: Reduced bulbs 960W

Overall Working Plain Illuminance 2





Reduced bulbs

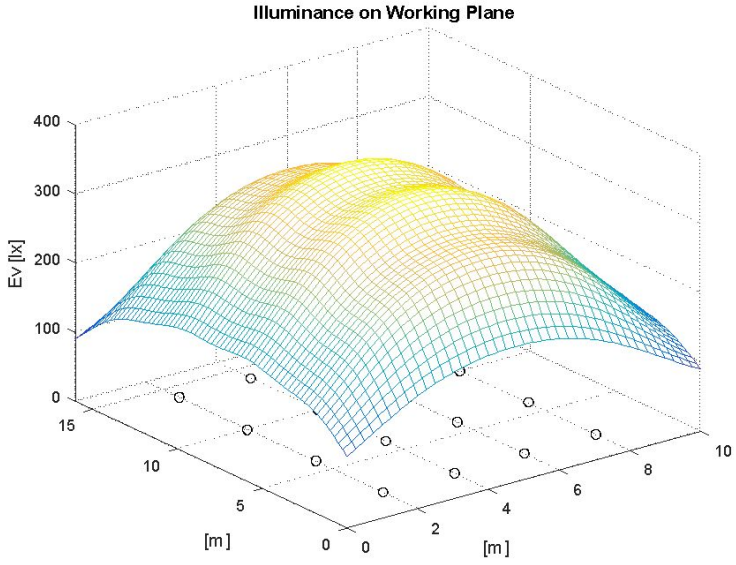
Total wattage: 896 W, saving: 128 W/13%

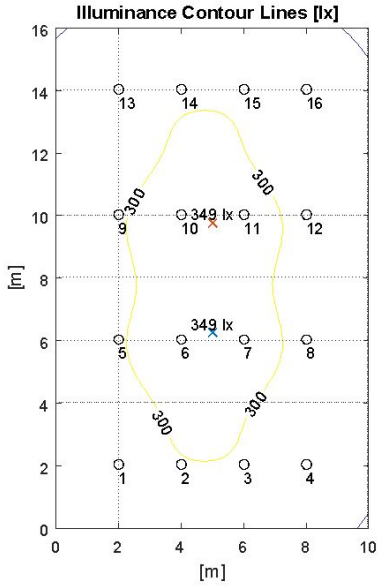
Light ID	Type	Class	Number of bulbs	Wattage each	Lumen(max) each
1	Fluorescent	W28T5	2	28	2900
2	Fluorescent	W28T5	2	28	2900
3	Fluorescent	W28T5	2	28	2900
4	Fluorescent	W28T5	2	28	2900
5	Fluorescent	W28T5	2	28	2900
6	Fluorescent	W28T5	2	28	2900
7	Fluorescent	W28T5	2	28	2900
8	Fluorescent	W28T5	2	28	2900
9	Fluorescent	W28T5	2	28	2900
10	Fluorescent	W28T5	2	28	2900
11	Fluorescent	W28T5	2	28	2900

12	Fluorescent	W28T5	2	28	2900
13	Fluorescent	W28T5	2	28	2900
14	Fluorescent	W28T5	2	28	2900
15	Fluorescent	W28T5	2	28	2900
16	Fluorescent	W28T5	2	28	2900

Table 4: Reduced bulbs 896 W

Overall Working Plain Illuminance 3





Reduced bulbs

Total wattage: 672 W, saving: 352 W/34%

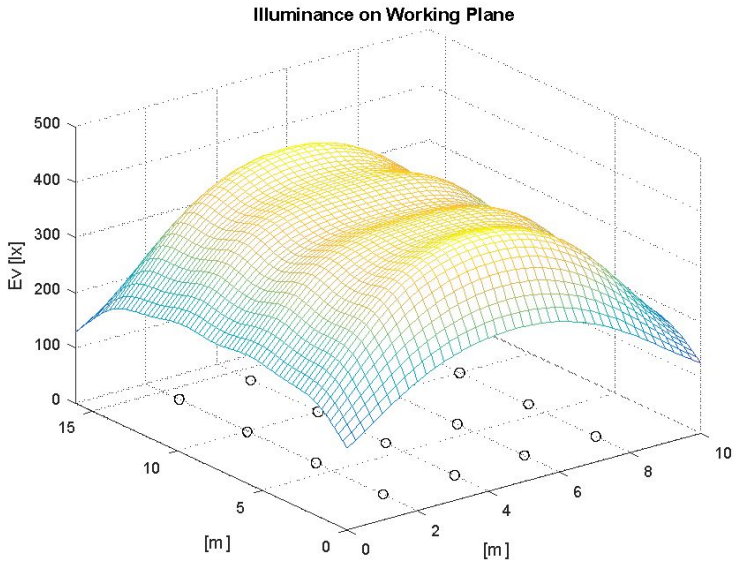
Light ID	Type	Class	Number of bulbs	Wattage each	Lumen(max) each
1	Fluorescent	W21T5	2	21	2100
2	Fluorescent	W21T5	2	21	2100
3	Fluorescent	W21T5	2	21	2100

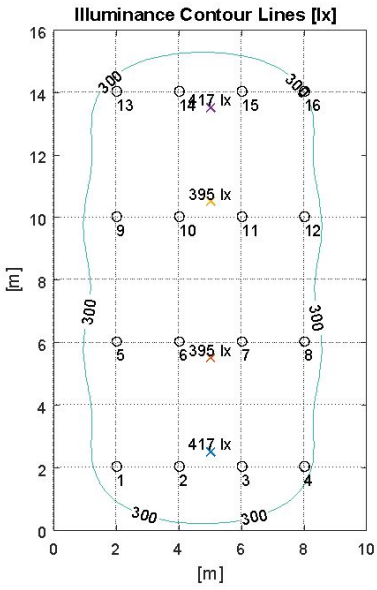
4	Fluorescent	W21T5	2	21	2100
5	Fluorescent	W21T5	2	21	2100
6	Fluorescent	W21T5	2	21	2100
7	Fluorescent	W21T5	2	21	2100
8	Fluorescent	W21T5	2	21	2100
9	Fluorescent	W21T5	2	21	2100
10	Fluorescent	W21T5	2	21	2100
11	Fluorescent	W21T5	2	21	2100
12	Fluorescent	W21T5	2	21	2100
13	Fluorescent	W21T5	2	21	2100
14	Fluorescent	W21T5	2	21	2100
15	Fluorescent	W21T5	2	21	2100
16	Fluorescent	W21T5	2	21	2100

Table 5: Reduced bulbs 672W

Delamped Lighting Plan

Overall Working Plain Illuminance 3





Delamped bulbs

Total wattage: 672 W, saving: 128 W/13%

Light ID	Type	Class	Number of bulbs	Wattage each	Lumen(max) each
1	Fluorescent	W32T8	2	32	3200
2	Fluorescent	W32T8	2	32	3200
3	Fluorescent	W32T8	2	32	3200
4	Fluorescent	W32T8	2	32	3200
5	Fluorescent	W32T8	2	32	3200
6	Fluorescent	W32T8	1	32	3200
7	Fluorescent	W32T8	1	32	3200
8	Fluorescent	W32T8	2	32	3200
9	Fluorescent	W32T8	2	32	3200
10	Fluorescent	W32T8	1	32	3200

11	Fluorescent	W32T8	1	32	3200
12	Fluorescent	W32T8	2	32	3200
13	Fluorescent	W32T8	2	32	3200
14	Fluorescent	W32T8	2	32	3200
15	Fluorescent	W32T8	2	32	3200
16	Fluorescent	W32T8	2	32	3200

Table 6: Delamped bulbs 672W